Wifi-Reports: Improving Wireless Network Selection with Collaboration

Jeffrey Pang, Ben Greenstein, Michael Kaminsky, Damon McCoy, and Srinivasan Seshan

Abstract—Wi-Fi clients can obtain much better performance at some commercial hot spots than others. Unfortunately, there is currently no way for users to determine which hot spot access points (APs) will be sufficient to run their applications before purchasing access. To address this problem, this paper presents *Wifi-Reports*, a collaborative service that provides Wi-Fi clients with historical information about AP performance and application support. The key research challenge in Wifi-Reports is to obtain accurate user-submitted reports. This is challenging because two conflicting goals must be addressed in a practical system: preserving the privacy of users' reports and limiting fraudulent reports. We introduce a practical cryptographic protocol that achieves both goals, and address the important engineering challenges in building Wifi-Reports. Using a measurement study of APs in a busy commercial district, we show that Wifi-Reports would improve the performance over previous AP selection approaches in 30-60 percent of locations.

Index Terms—Privacy, anonymity, wireless, reputation, 802.11.

✦

1 INTRODUCTION

USERS expect the Internet connectivity wherever they travel and many of their devices, such as iPods and wireless cameras, rely on local area Wi-Fi access points (APs) to obtain connectivity. Even smart phone users routinely employ Wi-Fi instead of 3G and WiMAX to improve the performance of bandwidth-intensive applications or to avoid data charges. Fortunately, there is often a large selection of commercial APs to choose from. For example, JiWire [6], a hot spot directory, reports 395-1,071 commercial APs in each of the top 10 US metropolitan areas. Nonetheless, users report that some APs block applications [9] and have poorer than advertised performance [25], so selecting the best commercial AP is not always straightforward.

1.1 Commercial Wi-Fi

To verify these reports, we present the first measurement study of commercial APs in hot spot settings. Previous wardriving studies [29], [33] 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 performance and application support of all visible APs at 13 hot spot locations in a busy commercial district over the course of one week. We find that there is indeed a wide range of AP performance even among APs very close to each other. Yet, there is currently no way for a user to determine which AP would be best to run his applications before paying for access.

1.2 Wifi-Reports

To address this problem, 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 hot spot database similar to existing hot spot directories but where APs are annotated with the performance information. Second, it enables users to select among APs visible at a location more effectively. Clients that participate in Wifi-Reports automatically submit reports on the APs that they use. Reports include metrics such as estimated backhaul capacity, ports blocked, and connectivity failures. Using submitted reports, the service generates summary statistics for each AP to predict its performance. Obtaining accurate user-submitted reports poses two challenges:

1. *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. 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 [12], even if they are tightly regulated like cell phone

- *J. Pang is with the School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213-3891. E-mail: jeffpang@cs.cmu.edu.*
- *B. Greenstein is with Intel Labs Seattle, 1100 NE 45th Street, 6th Floor, Seattle, WA 98105. E-mail: benjamin.m.greenstein@intel.com.*
- *M. Kaminsky is with Intel Labs Pittsburgh, 4720 Forbes Ave., Suite 410, Pittsburgh, PA 15213. E-mail: michael.e.kaminsky@intel.com.*
- *D. McCoy is with the Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, MC 0404, La Jolla, CA 92093-0404. E-mail: dlmccoy@cs.ucsd.edu.*
- *S. Seshan is with the School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213-3891. E-mail: srini@cmu.edu.*

records [4]. 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 [33] because it provides information on commercial APs that cannot be tested beforehand. In a few locations (30 percent), we found that Wifi-Reports even outperforms the strategy of picking the "official" AP for a hot spot. This may be because, for example, the AP next door has a better backhaul connection.

1.3 Contributions

1. To our knowledge, we are the first to study the attributes of commercial, encrypted, and "pay-for-access" APs in the wild. Although previous studies have examined open APs [29], [33] observed while war driving, we find that the best performing AP for a typical user in one commercial district is most often a closed AP.

2. We show that Wifi-Reports' summary statistics predict the 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 percent 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. Unlike previous AP selection approaches, Wifi-Reports enables users to examine the characteristics of APs that are not in radio range, which is useful when users are mobile.

3. We present the design, implementation, and evaluation of a practical protocol that enables users to contribute reports on APs anonymously, and generates accurate summary statistics for each AP even if 10 percent of that AP's users collude to promote it. Although we use this protocol in the context of Wifi-Reports, it is applicable to other collaborative reporting services.
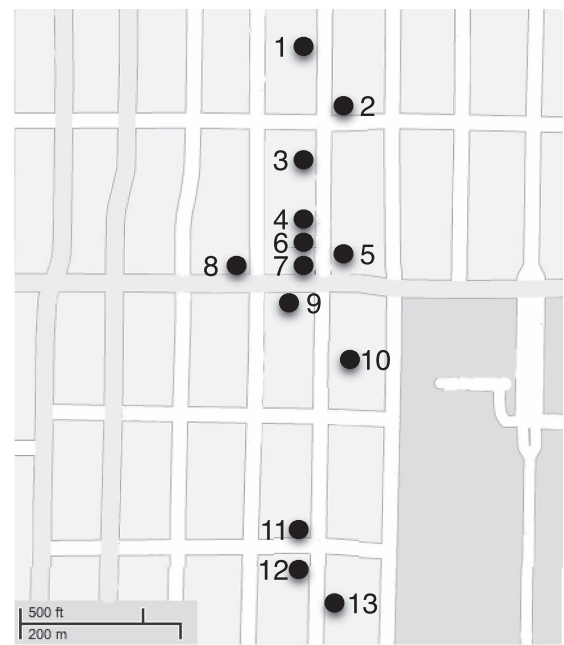

Fig. 1. Proximity of measured hot spot locations.

The rest of this paper is organized as follows: Section 2 presents the results of our measurement study. Section 3 presents an overview of Wifi-Reports' design. Section 4 describes how it preserves privacy and mitigates fraud. Section 5 describes how it distinguishes client locations. Section 6 presents an evaluation of Wifi-Reports. Section 7 presents related work, and Section 8 concludes.

2 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 performance and application support of different hot spots' APs? The more diversity, the more likely a user will choose a hot spot with substantially suboptimal performance when selecting randomly from a hot spot 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 performance predictable enough so that historical information would be useful?

Our study examined hot spots in a busy commercial district near a university. We believe that 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. Fig. 1

shows the hot spot 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 hot spot's official AP, the APs of hot spots nearby are also usually visible. APs of the free public network that we call public-net are sometimes visible at all locations. APs belonging to a university network that we call university-net are sometimes visible due to proximity to campus, 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.

2.1 Setup

2.1.1 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 [29].

2.1.2 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 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. We only measure infrastructure networks (i.e., we do not measure any ad hoc networks).[1]

We performed measurements at typical seating locations in each hot spot. Although the exact same location was not used for all measurements in a hot spot, Section 5 shows how well we can distinguish the performance at different locations.

2.1.3 Time Frame

Previous studies measured each AP at a single point in time [29], [33]. Since we want to know whether AP characteristics are predictable, we performed 8-13 measurements at each location (with the exception of cafe 6, where we only performed six trials). These measurements were taken during seven 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 hot spot opened later or closed earlier).

2.2 Results

2.2.1 Basic Connectivity

Fig. 2a shows the fraction of times we were able to obtain connectivity from each AP at each location (i.e., as 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 three times and for up to 30 seconds on failure, which ever comes first). 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 hot spot is shown in a solid color and its symbol is in a larger font.[2]

As expected, most (10 of 13) official hot spot APs were successful 100 percent of the time. Some, however, such as the APs at cafe 2 and cafe 13, failed several times. These were all DHCP failures and frequent users of cafe 13 say that the AP has always had DHCP problems. However, it would be difficult to detect these problems automatically because even attempting to access the network requires a WPA password from the cashier. Although unofficial APs visible at hot spots tended to fail with higher regularity due to wireless loss, a few APs in most locations (8 of 13) succeeded whenever they were visible in our network scan. Thus, even this very basic connectivity metric suggests that there is diversity.

2.2.2 TCP Throughput

Adequate throughput is important for many applications, such as streaming video or VoIP. Figs. 2b and 2c show box-plots of the TCP download and upload throughputs achieved through each AP, respectively (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). We measured throughput over the final 5 seconds of a 10-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 hot spot locations. Median download capacities range from less than 100 Kbps (e.g., an AP at cafe 6) to over 5 Mbps (e.g., APs at cafe 3 and cafe 5), and median upload capacities range from 50 kbps to over 3 Mbps (e.g., APs at cafe 8). There is variability in each AP's throughput, 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 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 hot spot APs have the highest median throughputs at their respective locations. However, this is not true at cafe 2, cafe 3, cafe 6, and cafe 8, where better APs were available from an apartment building next door, the public-net network, a store next door, and a nearby hotel, respectively. Indeed, at cafe 3 and cafe 6, an unofficial

---

1. One physical AP at cafe 3 and cafe 12 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 these hot spots are migrating from one ISP to another.

2. cafe 13 has two official APs because it changed APs in the middle of our measurement period. However, both APs suffered from basic connectivity problems.
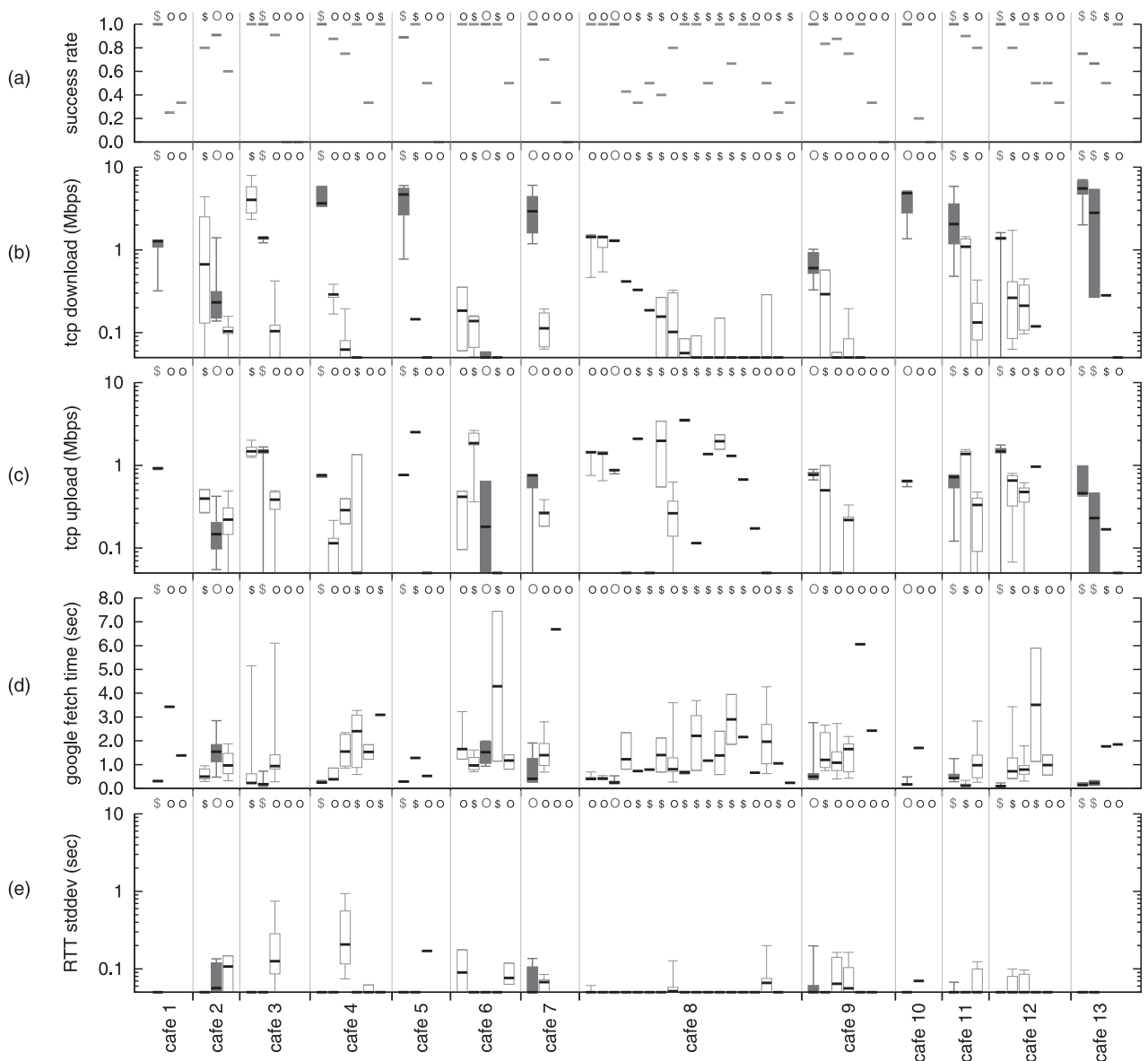
Fig. 2. (a) The success rate of different APs (i.e., how often we could connect and access the Internet when each AP was visible). Each point represents one AP visible at each location. (b) A box-plot of the measured TCP download throughput through each AP. Note the logarithmic scale. (c) A box-plot of measured TCP upload throughput. (d) A box-plot of the time to fetch http://www.google.com using each AP. (e) A box-plot of RTT variance (in standard deviations from the mean) to the measurement server. The measurements for each AP are grouped by the hot spot location where they were taken, as 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 hot spot 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 nonfree APs at cafe 8 are university-net APs in a building across the street.

AP always had significantly higher throughput than the official one when visible. Recall that these comparisons only include measurements when we were able to successfully pay for and obtain the Internet connectivity, so a user without historical information would have to pay before discovering this.

2.2.3   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 websites. Fig. 2d 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 6 KB, fetch time is dominated by latency rather than transfer time. We do not count the times when association failed.

Just as 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 hot spot locations, but this is not true at cafe 2, cafe 6, and cafe 11. Only the disparity between the best and official APs at cafe 2 is large enough to be noticeable, but

3. The CNAME and A DNS records for www.google.com have TTLs of one week and one day, respectively, so they are almost always already cached.

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 cafe 3), so ranking is dependent on application requirements. Finally, all official APs, except the one at cafe 7, 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.

2.2.4 Delay Jitter

High variation in response time can disrupt applications that rely on a steady stream of packets, such as streaming video. Fig. 2e shows a box-plot of network jitter in terms of the standard deviation of round-trip times to a well-connected measurement server under our control. RTTs were measured using ICMP pings. Very few APs exhibit high jitter, though some exhibit higher variance than others. For example, the official APs at cafe 2 and cafe 7 have RTT standard deviations over 100 ms at least 25 percent of the time. Most other official APs never have noticeable variation, and thus, would be more suitable for applications that are sensitive to jitter. In addition, an unofficial AP at cafe 2 is present with consistently lower jitter.

2.2.5 Port Blocking

To determine if an AP blocked or redirected certain application ports, we sent three 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, each probe tried to establish a connection and download $\sim$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 (cafe 10's official AP) blocked more ports: all nonprivileged 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.

2.2.6 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 percent of hot spot locations, so ranking APs is not always obvious. Finally, with respect to *predictability*, there is variability in the 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.

2.3 Discussion

2.3.1 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 percent

of locations. First, in Section 6.1, we show that historical information can get us the best AP in the remaining 30 percent. Second, as hot spot 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.

2.3.2 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 in this paper 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.

3 WIFI-REPORTS OVERVIEW

Wifi-Reports is a recommendation system [15]. 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.

3.1 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 [36]. 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 [18], [28]. Nevertheless, we still want to limit the influence of malicious users who submit fraudulent reports, which is a common problem in recommendation systems [42], [44].

Location context. Clients will typically search for summaries by location (e.g., "all APs in San Francisco"), 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.

3.2 System Tasks

The operation of Wifi-Reports consists of three main tasks (Fig. 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.

3.2.1 Measure and Report

Clients measure and submit reports on APs that they use. For example, suppose a client attempts to connect to the Internet
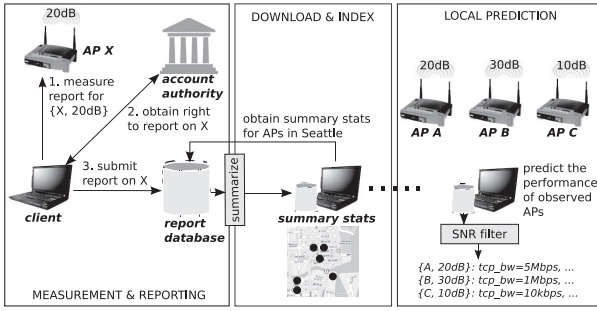
Fig. 3. Wifi-Reports components and typical tasks.

using AP X. If the connection fails (i.e., association, DHCP, or all TCP connections fail), the client generates the report $\{\text{ap} = \mathbf{X}, \text{SNR} = 20 \text{ dB}, \text{date} = 11/14/2008, \text{connectivity} = \text{false}\}$. [4] If the connection succeeds, 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

$$\{\text{ap} = \mathbf{X}, \text{SNR} = 20 \text{ dB}, \text{date} = 11/14/2008, \text{connectivity}$$
$$= \text{true,tcp\_bw\_down} = 100 \text{ kbps}, \text{google\_resp\_time}$$
$$= 500 \text{ ms}, \ldots\}.$$

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.

3.2.2 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 $\{0, 1\}$ 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.

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 [5] client.

4. $\mathbf{X}$ refers to the AP's BSSID and a hash of its signing key described in Section 4.

5. A number of techniques and tools exist to estimate bandwidth [37] and response time [3]. These techniques are outside the scope of this paper, but the measurements we used can be implemented as an anonymous speed test.

To find a suitable hot spot, reports are shown to a user on a map. In order to facilitate this operation, reports must be searchable by geographic location. Unfortunately, we cannot rely on GPS because many clients are not equipped with it and it often does not work indoors. We describe existing techniques that we leverage to obtain coarse geographic coordinates in Section 5.1.

3.2.3 Predict Locally

Finally, when a user sits down at a cafe, he typically wants to find the best visible AP. 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 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 5.2.

4 LOCATION PRIVACY

This section describes a novel report submission protocol that ensures *location privacy* and *limited influence*, and 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, $u = \text{submitter}(R)$ to be the user that submitted report R, and $s = \text{target}(R)$ 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 $\text{submitter}(R_i)$ with probability greater than $\frac{1}{|U \setminus C|}$, for all reports R_i. 2) No one should be able to 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 $\text{submitter}(R_i) = \text{submitter}(R_j)$ with probability greater than $\frac{1}{|U \setminus C|}$, for all R_i, R_j, where $\text{submitter}(R_i) \neq \text{submitter}(R_j)$ or $\text{target}(R_i) \neq \text{target}(R_j)$. 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 $\text{target}(R_{k+1})$ 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 $\text{submitter}(R_i) = \text{submitter}(R_j)$ and $\text{target}(R_i) = \text{target}(R_j)$, anyone *should be able* to verify that $\text{submitter}(R_i) = \text{submitter}(R_j)$. 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.

4.1 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., [43]). 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 6.3 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 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 [24] and that the mix achieves its goal, i.e., no one can infer the source IP address of the sender's messages.

4.2 Straw Man Protocols

4.2.1 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 cannot distinguish when one user submits many reports on an AP versus when many users submit one report each on the AP.

4.2.2 Authenticate Reports

To provide limited influence, 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 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 [17], 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 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.

4.3.1 Preliminaries

The RSA blind signature scheme [17] is a well-known cryptographic primitive that we use in our protocol. Let $\mathrm{blind}(K, m, r)$ and $\mathrm{unblind}(K, m, r)$ be the RSA blinding and unblinding functions using RSA public key K, message m, and random blinding factor r (we use 1,024-bit keys and values). Let $\mathrm{sign}(K^{-1}, m)$ be the RSA signature function using RSA private key K^{-1}, and let $\mathrm{verify}(K, m, x)$ be the RSA verification function, which accepts the signature x if and only if $x = \mathrm{sign}(K^{-1}, m)$. Let $H(m)$ be a public pseudorandom hash function (we use SHA-512). We leverage the following equivalence:

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

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, $\bar{m} = \mathrm{blind}(K, m, r)$ 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

$$
\begin{aligned}
A: \quad & \{M, M^{-1}\}, \{M_j, M_j^{-1}\} \; \forall s_j \in S, \\
& msig_j \leftarrow \mathsf{sign}(M^{-1}, H(bssid_j | M_j)) \; \forall s_j \in S \\
u_i: \quad & M, M_j, msig_j \\
u_i: \quad & \{K_{ij}, K_{ij}^{-1}\}, r \xleftarrow{R} \{0,1\}^{1024} \quad\quad (1) \\
u_i: \quad & b \leftarrow \mathsf{blind}(M_j, H(K_{ij}), r) \quad\quad (2) \\
u_i \to A: \quad & \texttt{"sig-request"}, s_j, b \quad\quad (3) \\
A: \quad & sig'_{ij} \leftarrow \mathsf{sign}(M_j^{-1}, b) \quad\quad (4) \\
A \to u_i: \quad & \texttt{"sig-reply"}, sig'_{ij} \quad\quad (5) \\
u_i: \quad & sig_{ij} \leftarrow \mathsf{unblind}(M_j, sig'_{ij}, r) \quad\quad (6)
\end{aligned}
$$

Fig. 4. GENTOKEN(u_i, s_j).

$x_{k+1} = \mathsf{sign}(K^{-1}, H(\hat{m}_{k+1}))$ 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 [17]. However, anyone can check whether $\mathsf{verify}(K, H(\hat{m}_i), x_i)$ accepts.

4.3.2 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.

4.3.3 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). Fig. 4 shows the formal protocol but we detail each step below.

Before the protocol begins, a number of elements have already been generated and exchanged when a client initially signs up for the service. These items are listed in the lines before step 1 in Fig. 4. More specifically, 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).

As illustrated in Fig. 4, the GENTOKEN(u_i, s_j) protocol exchange has six steps:

1. First, u_i generates a new *reporting* key pair $\{K_{ij}, K_{ij}^{-1}\}$ and a 1,024-bit random value r known only to u_i. The key pair will be used by u_i to sign reports on s_j. r is used as a secret random input to the blinding function.

2. The authority A must authorize the public key K_{ij} so that other parties will know that it is valid. However, u_i does not want to reveal K_{ij} to A. Therefore, u_i computes the hash $H(K_{ij})$, which securely and uniquely identifies K_{ij}, and blinds it using the blinding function of the blind signature protocol. By blinding using M_j, the authority A can sign using its corresponding private key M_j^{-1}, the signing key for AP s_j.

$$
\begin{aligned}
D_k: \quad & M, M_j \; \forall s_j \in S \\
u_i: \quad & rsig \leftarrow \mathsf{sign}(K_{ij}^{-1}, H(R)) \quad\quad (7) \\
u_i \to D_k: \quad & \texttt{"report"}, s_j, K_{ij}, sig_{ij}, R, rsig \quad (8)
\end{aligned}
$$

Fig. 5. SUBMITREPORT(u_i, s_j, R).

3. User u_i sends the blinded hash b to the authority, along with s_j, which identifies the AP it is for. A then checks whether it has already sent a `sig-reply` message to u_i for s_j. If so, it aborts because we must ensure that u_i only receives one key per AP; otherwise, it continues.

4. A signs the blinded hash b using M_j^{-1}, the signing key for AP s_j. The resulting sig'_{ij} is the blinded signature of the original hash. By blindness, A does not learn K_{ij} only that it is generating a key for s_j. Thus, no one can link K_{ij} to user u_i or to any $K_{il}, l \neq j$. This is the central step to protect location privacy.

5. A sends the blinded signature sig'_{ij} to user u_i.

6. Finally, the user u_i unblinds the blinded signature sig'_{ij} to produce sig_{ij}. The user then checks that $\mathsf{verify}(M_j, H(K_{ij}), sig_{ij})$ accepts. If it does (i.e., the authority followed the protocol), then sig_{ij} is a valid signature of the hash $H(K_{ij})$. u_i saves $K_{ij}, K_{ij}^{-1},$ and sig_{ij} for future use.

$\{K_{ij}, sig_{ij}\}$ 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.

4.3.4 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 (see Fig. 5):

7. First, user u_i signs a hash of the report R using his reporting key K_{ij}^{-1} for the target AP s_j. The signature $rsig$ along with the hash $H(R)$, which uniquely identifies R, prove to third parties that R is a report that could only have been generated by user u_i for AP s_j.

8. The user sends these items to the report databases. In addition, the message includes K_{ij} and sig_{ij}, the public signing key for s_j and the authority's signature on it, respectively. These items enable anyone to 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. The `report` message is sent through a mix network, so it does not explicitly reveal its sender. After step 8, each database 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 fails, the report is invalid and is discarded.

4.3.5 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 $\frac{1}{|S|}$, and 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 that 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 6, we show that using the granularity of a city is practical.[6]

4.3.6 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 adds an AP identified by BSSID and located via beacon fingerprint (see Section 5.1). A generates a new signing key pair and its signature $\{M_j, M_j^{-1}\}$, $msig_j \leftarrow \text{sign}(M^{-1}, H(bssid_j|M_j))$, 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 hot spot 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 reports. An AP is removed from S if it is not reported on in three 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 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.

4.3.7 Handling Long-Term Changes

AP performance can change over time due to backhaul and AP upgrades. However, these changes typically occur at timescales of months or more. Thus, reports have a time-to-live (TTL) of three 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. The TTL is set by the client at the time of report submission, so it can independently determine when its own reports expire.

4.3.8 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 5.2).[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. Recall that a client sets the TTL on its reports, so it can independently determine the expiry time. 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]

4.3.9 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 straightforward 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 8), user u_i sends $h = \text{blind}(M, H(report), r)$ to A. If u_i has not already exceeded t reports this week, A sends $lsig' = \text{sign}(M^{-1}, h)$ back to u_i, and u_i unblinds $lsig'$ to obtain $lsig = \text{sign}(M^{-1}, H(report))$. $lsig$ 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.

The values 10-20 would be reasonable for t; analysis of Wi-Fi probes show that most clients have not used more than 20 APs recently [27]. This approach only adds 4 ms of computational overhead on A per report submitted (see Section 6.2).

4.4 Security Analysis

We now describe how Wifi-Report's reporting protocol maintains location privacy, limited influence, and service availability in the face of attacks. We first describe how the

---

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.

7. A more sophisticated summarization algorithm might use the mean or median values of all a user 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 $\{\text{"revoke"}, u_i, K_{ij}, ksig\}$ to A and the databases, where $ksig \leftarrow \text{sign}(K_{ij}^{-1}, H(\text{"revoke"}|u_i|K_{ij}))$, which proves that u_i has K_{ij}'s secret key and K_{ij} (and all reports signed with it) is revoked.

protocol prevents report linking and forging attacks in general, and then, discuss the specific impact of a dishonest account authority, dishonest databases, dishonest APs, and dishonest users.

4.4.1 Report Linking Attacks

To preserve location privacy, it must not be possible to link a user to his reports or his reports to one another. To achieve this, we must ensure two properties: First, there must be no explicit information in a submitted report that can be used to link it to a user or another report. Second, a report must not be linked to the submitting user during the report submission process. We consider each of these properties in turn. First, recall that a submitted report contains "report", $s_j, K_{ij}, sig_{ij}, R, rsig$ (see step 8 in the protocol). Neither the contents R nor the target AP s_j reveals any explicit information that can be used to link it to a user or another report. $rsig = \text{sign}(K_{ij}^{-1}, H(R))$ does not reveal any linking information because K_{ij}^{-1} is only known to the submitter and is a different random value for each AP. Therefore, it suffices to show that $\{K_{ij}, sig_{ij}\}$ cannot be linked to user i or any other $\{K_{ik}, sig_{ik}\}$, for all $j \neq k$. No party other than A and the report databases receive any information about K_{ij} (which is random) or sig_{ij} (which is derived from K_{ij}). A only receives information derived from K_{ij} during the GENTOKEN phase. Thus, we simply require that A cannot link K_{ik} or sig_{ik} back to a particular user during the GENTOKEN phase and no database can link K_{ik} or sig_{ik} back to a particular user during the SUBMITREPORT phase. The first requirement is ensured because the user blinds K_{ij} before submitting it to A, and by the blindness property of the RSA blind signature protocol, neither it nor its unblinded signature reveals any information about their contents to A. The second requirement is ensured because users submit reports through a mix network to the report databases during the SUBMITREPORT phase, which prevents the DBs from explicitly linking any received reports to the submitting user.

Side-channels exposed in R and the time it is submitted 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 time stamp 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 time stamps and submission times, we believe that we can defeat most of these attacks in practice without significantly degrading accuracy.

4.4.2 Report Forging Attacks

To ensure limited influence, we assume that users can only obtain one signed token for each AP in the system. This property ensures that only one report from each user will be included in an AP's summary statistics because reports without a valid signed token will be discarded. We must ensure that there is only one token per user even if multiple users collude. Therefore, to violate limited influence, N users would have to obtain $M > N$ tokens for a single AP. Assuming that A obeys the protocol and does not give out multiple tokens per user, these N users

must be able to forge a new validly signed token from their N existing tokens to obtain more tokens. But if they could do this, then the unforgeability property of the blind signature protocol would be violated.[9]

4.4.3 Dishonest Account Authority

A dishonest account authority may try to link a user to his reports by giving a token to him but no one else. For example, 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. Thus, clients can detect if A does not provide them with a token that they should have received. They can then protect their location privacy by refusing to use the service.

While Wifi-Reports ensures that A cannot violate a user's location privacy, it does not prevent the account authority from manipulating an AP's summary statistics. For example, because A is responsible for signing tokens, it can "mint" as many as it desires. As with existing high-profile online reputation systems, we assume that the authority is disincentivized to manipulate summary statistics because it wants to be a useful service to its users (i.e., customers). One technical measure we could take to mitigate result manipulation would be to use multiple independent authorities and require each report to be signed with one token from each. Then, all authorities would have to collude to carry out a minting attack.

4.4.4 Dishonest Report Databases

A dishonest database cannot forge valid reports, but it can delete valid reports that it receives before their expiry and deny particular clients access to reports. However, if the majority of the databases are honest, then clients can detect dishonest databases when their results differ from that of the majority. Even if only one report database is honest, users can audit the remainder by requesting all their reports. Since databases can only alter results by dropping reports, the database with the most complete set of reports will be correct.

4.4.5 Dishonest APs

Dishonest APs may try to hijack the reputation of other APs. In addition, their interaction with clients might serve as a side-channel to link clients to reports.

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

---

9. We note that it is important that A signs $H(K_{ij})$ rather than K_{ij} directly. Otherwise, two colluding users a and b could forge a third valid token from the two that they have. This is because $sig_{fj} = sig_{aj} \cdot sig_{bj} \bmod N_j$ is a valid signature on some value. If signatures were derived directly from the token K_{ij}, then this value could be used as a token as well. By deriving the signature from $H(K_{ij})$ instead, in order for an adversary to use the new value as a valid token, he would have to invert the hash function $H(\cdot)$, which is computationally infeasible.

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., [11]) may be a sufficient deterrent in practice.

AP reidentification attacks. An AP may be able to identify which reports about it were submitted by each of its users with the following attack: it provides each user a different quality of service so that the quality of service reported can be used to implicitly reidentify the user. However, the AP already knows when the user visits because they log in, so no further location privacy is violated. We believe that the revelation of a user's performance statistics is not sensitive enough for concern, since the AP can learn this information on its own.

4.4.6 Dishonest Users

Dishonest users may try to collude to change an AP's reputation, clog the system with imaginary APs, and attempt to introduce inconsistency in databases by only submitting reports to a subset.

Large-scale collusion attacks. If a large number of clients collude, such as compromised hosts participating in a botnet, they can potentially alter the reputation of a target AP. If the number of malicious users M submitting reports on an AP s_j is greater than the number of honest users H submitting reports on s_j, then the summary statistics for s_j will be skewed. Although it is reasonable to assume that most users are honest, honest users will only report on APs in locations where they visit. In contrast, malicious users can report on any AP, whether they are near it or not. In order to make it more likely that $M < H$, we should prevent malicious users from reporting on APs that are not near them; this would ensure that all malicious users reporting on an AP are drawn from the same population as honest users reporting on that AP. One way to accomplish this is to require users to attach a "proof" of their location to submitted reports.

A client can anonymously obtain a "proof of location" as follows: before submitting a report R, the client anonymously connects to the account authority (i.e., without authentication), which geolocates the client's IP address to the granularity of a city k. Assume that the authority has one location key pair $\{L_k, L_k^{-1}\}$ for each city k. The client sends a blinded hash if its report $b \leftarrow \text{blind}(L_k, H(R), r)$ to the authority using a new random value r. The authority signs b with L_k^{-1} and returns $sig'_b \leftarrow \text{sign}(L_k^{-1}, b)$ to the client. The client unblinds sig'_b to obtain $sig_b \leftarrow \text{unblind}(L_k, sig'_b, r)$, which is a valid signature on $H(R)$ using the location key L_k^{-1}. These three steps comprise the same blind signature protocol described in Section 4.3. Blinding ensures that the

authority does not know what it signed or who it came from, only that it came from an IP address in the city k. By attaching $\{L_k, sig_b\}$ to the submitted report, each database can then check whether the AP reported on by the report is in the city k, ensuring that only a client in k can report on APs in k.

Of course, a foreign client could tunnel such location proof requests through a compromised client on a local IP. Therefore, the authority rate limits the number of location proofs that a single IP can receive (e.g., 1 per day). This limiting means that an adversary can only obtain as many location proofs per day as he has compromised clients in the target city; compromising foreign hosts does not help. Although some IPs may be shared between multiple honest clients, most clients will likely have at least one that is not shared, such as at their home. Moreover, they can save reports until they obtain a valid location proof.

AP addition spamming. Wifi-Reports cannot directly verify the validity of the APs that users add. A malicious user could spam the system with the addition of many fake APs, forcing users to download unnecessary tokens. This does not affect Wifi-Report's correctness, but degrades the performance. The account authority mitigates this attack by rate limiting the addition of APs. Since fake APs in the system will expire after a fixed time if not reported on, and malicious users' reports are also rate limited, the total number of fake APs that a fixed number of malicious users can maintain in the system is constant.

Incomplete report submission. A malicious user might try to flag a database as dishonest by submitting its reports to all but that database. Databases avoid this attack by periodically requesting new reports from each of the other report databases. Users and databases should query databases through a mix network (or a random third-party proxy), lest a malicious database colluding with a malicious user attempts to deny reports to a database but provide them to real users.

5 LOCATION CONTEXT

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

5.1 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 [32]. Existing war-driving databases is sufficient to facilitate this task (e.g., Skyhook [7] 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 paper. We note that coordinates are only needed to allow clients to search for AP summary statistics by location.

5.2 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 multipath interference [30]. 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 percent loss, a range where clients experience intermediate loss, and a range where clients experience near 0 percent loss [30]. 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 [30], 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.[10] We assume that reports of connectivity failures experienced 100 percent 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 percent loss ranges, so we put all measurements in a single range.

Fig. 6 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
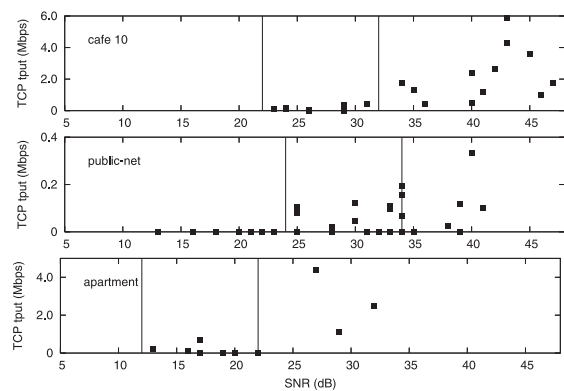

Fig. 6. Estimated 100 percent, intermediate, and 0 percent loss regions for three APs in our measurement study.

reports in the 0 percent loss region with the 100 percent 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 [41] or by actively probing each AP.

5.3 Discussion

5.3.1 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, 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 [30]. 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 the accuracy, we can leverage existing self-calibration techniques that determine the biases of sensors (e.g., [16]). Implementing a distributed calibration algorithm is the subject of future work.

5.3.2 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 multipath interference and wireless contention [39], [41]. However, we found that contention is rare in our measurement study, so prediction accuracy is good even discounting these factors (see Section 6).

5.3.3 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 is 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.

6 EVALUATION

We evaluate the utility and practicality of Wifi-Reports using our measurement study (see Section 2) and our

---

10. When we have more than a few samples (i.e., ≥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.

implementation of the reporting protocol (see Section 4). 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?

6.1 AP Selection Performance

6.1.1 Setup

We use our measurement study to simulate two scenarios: First, we evaluate the scenario where a user chooses which hot spot to go to physically based upon the predicted performance of all hot spots nearby. In this scenario, a user is primarily interested in *prediction accuracy*, i.e., we want $\text{predict}(s)/\text{actual}(s)$ to be close to 1 for each AP s, where $\text{predict}(s)$ is the predicted performance (e.g., throughput) of s and $\text{actual}(s)$ 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 [33], [39], [41], i.e., given the set of visible APs $V = \{s_1, s_2, \ldots, s_n\}$, we want a selection algorithm $\text{select}(\cdot)$ that maximizes $\text{actual}(\text{select}(V))$, where $s = \text{select}(V)$ 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 $\text{actual}(\text{select}(V))/\max_{s \in V}(\text{actual}(s))$. In Wifi-Reports, $\text{select}(V) = \text{argmax}_{s \in V}(\text{predict}(s))$.

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, $\text{actual}(s)$ is equal to one of our measurements of s. We evaluate the performance over all our measurement trials. To simulate the $\text{predict}(s)$ 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 $\text{predict}(s)$. 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.

6.1.2 Prediction Accuracy

Fig. 7 shows CDFs of prediction accuracy over all trials of official hot spot APs for TCP download throughput and
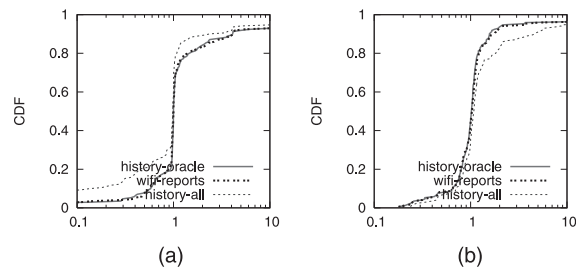

Fig. 7. CDF prediction accuracy for (a) TCP download throughput and (b) Google fetch time over all trials on all official APs at their respective hot spots. Note the logarithmic scale on the x-axis.

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 hot spot 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 5). 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 cafe 12, 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 percent 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 percent 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 percent 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 percent loss region. Furthermore, we note that at least 75 percent of predictions for both metrics are within a factor of 2 of the achieved value, while Fig. 2 shows that the difference in the median throughputs and response times of official APs can be up to 50 times and 10 times, respectively. Therefore, most predictions are accurate enough to make correct relative comparisons.

6.1.3 Ranking Accuracy

We now examine the scenario when a user is choosing between APs at a single location. Figs. 8a and 8b show boxplots 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 [33], an algorithm that associates with and
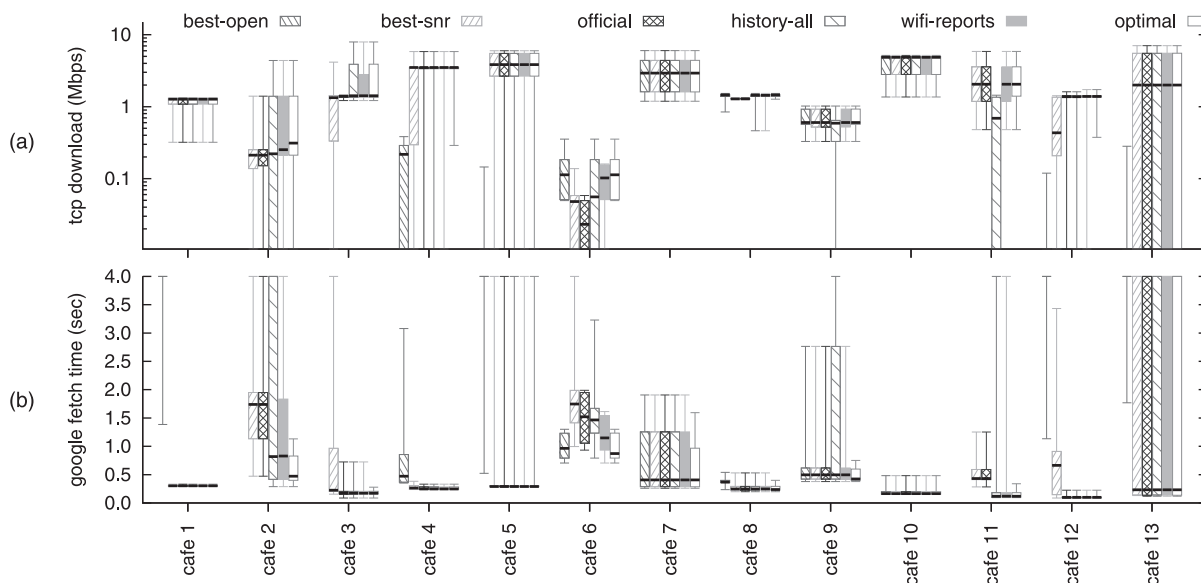
Fig. 8. (a) Box-plot of achieved TCP download throughput when using each of five AP selection algorithms at each location. Note the logarithmic scale. Missing boxes for the best-open algorithm are at 0. (b) Box-plot of the achieved response time of http://www.google.com using each of five AP selection 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 is ordered in the same order as the key at the top.

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 2 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 two 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 cafe 2, cafe 3, and cafe 11 since no open APs were visible. best-open performs better than all other algorithms only at cafe 6, where most of the APs were open. We note that best-open is qualitatively different from 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.

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 percent of the time at cafe 2, cafe 9, and cafe 11.

In contrast, wifi-reports achieves the performance closest to optimal for both metrics in all cases except for two. It achieves worse TCP throughput than best-open once at cafe 6 and worse response time than best-snr or official once at cafe 11. 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 often than all other algorithms at cafe 3 and cafe 6 and better response time more often than all other algorithms at cafe 2 and cafe 11. 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 APs that are nearby but not visible. This is useful when users are willing to move to obtain better connectivity.

6.2 Report Protocol Performance

We implemented our reporting protocol (Section 4) 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.

6.2.1 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 3.4 GHz 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.

6.2.2 Processing Time

Table 1 presents microbenchmarks of each step of the protocol. All times are in milliseconds. The most heavy-weight steps are the generation of 1,024 bit RSA keys by

TABLE 1
Microbenchmarks of Cryptographic Processing Times

	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

All keys are 1,024-bit RSA keys and SHA-512 is used as the hash function. All values are in milliseconds with a resolution of 10 microseconds. One thousand trials were executed.

both the client (K_{ij}) and server (M_j).[11] However, both keys can be generated anytime beforehand, so these operations need not be executed inline in the GENTOKEN 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 5,000 times per second.

6.2.3 Token Fetch Time

A user who wants to obscure his locations within a region must perform GENTOKEN on all APs in that region. Fig. 9a shows the end-to-end time to fetch tokens for all APs in each of the 10 cities that JiWire [6] 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 hot spots 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 [10] is a database of all APs that war drivers have overheard, including both commercial and private APs. Fig. 9b 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. Fig. 9c shows the CDF of the number of WiGLE APs in 1 km^2 areas in each of the cities. Most cells have fewer than 188 APs, which only take about 1 second to fetch, and no cell has more than 7,400, 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 the performance.

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

6.2.4 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.)

6.2.5 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 Fig. 9, chosen at random weighted by each city's population (according to 2007 US census data [40]). 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 $27,455x$ 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 [1], this would only cost about 0.02 cents per user for CPU and bandwidth resources. For all WiGLE APs, the authority can admit $165x$ 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 Section 4.3.9). However, these costs are also trivial. For example, even if 10 new hot spots 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.

6.3 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.

6.3.1 Setup

We consider the same scenario as in Section 6.1. 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-percent-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
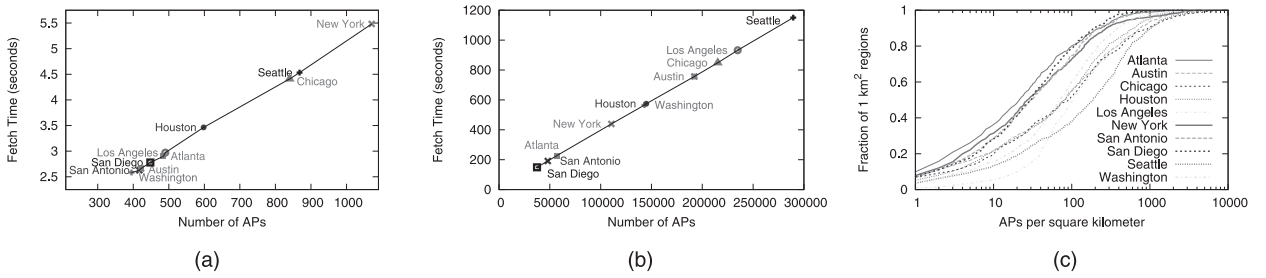
Fig. 9. (a) Time to acquire the right to report on all APs listed by JiWire in the top 10 cities. (b) Time to acquire the right to report on all APs listed by WiGLE in each of the same 10 cities. (c) CDF of the number of APs listed by WiGLE in each 1 km^2 region of a $32 \text{ km} \times 32 \text{ km}$ grid centered on each of 10 cities.

fraudulent reports described above and $100 - x$ are honest reports that are randomly sampled (with replacement) from our ~10 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 6.1. For comparison to Fig. 7a, we again focus on official APs.

6.3.2 Accuracy

Fig. 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 percent of reports are fraudulent. Even with 30 percent of fraudulent reports, most predictions are still correct within a factor of 2. However, when 50 percent of reports are fraudulent, most predictions are gross overestimates. This result is expected since the median function is not robust to 50 percent or more outliers larger than the actual median.

6.3.3 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-percent-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 that some honest users will eventually report on these APs and correct their summary statistics.

6.4 Application to Other Crowd-Sourced Services

Although we presented our reporting protocol in the context of a hot spot 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 divide items into subsets that are at most hundreds of thousands in size and do not reveal sensitive information (e.g., one subset for all items in each city). Second, the protocol cannot be applied directly when personalized collaborative filtering is required.

6.4.1 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 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 much larger, then the cost of generating tokens becomes prohibitive.

Fig. 11 shows the per-user cost and the total download time as functions of the number of tokens each user downloads. To estimate server costs, we use the same analysis as in the previous section based on EC2 CPU and bandwidth costs. We show the minimum possible download times given sufficient server resources, that is, assuming that
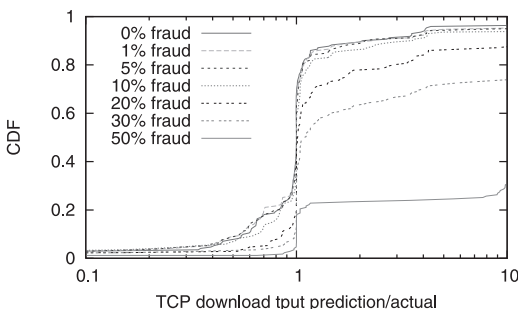

Fig. 10. CDF of prediction accuracy for TCP download throughput of all official APs at their respective hot spots. We vary the percentage of fraudulent reports that claim throughput is 54 Mbps. Note the logarithmic scale on the x-axis.
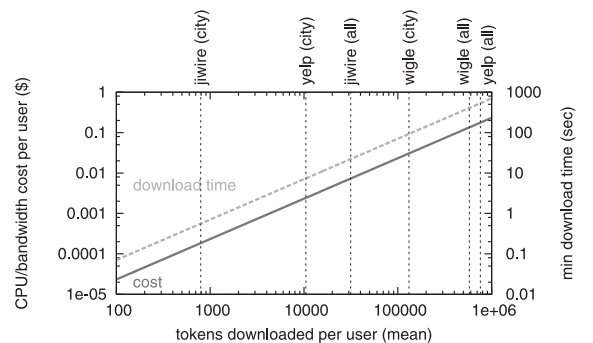

Fig. 11. The per-user cost and the total download time as functions of the number of tokens each user downloads. Costs are estimated based on EC2 CPU and bandwidth costs and download times assume that the bottleneck is a client on a typical cable modem (6 Mbps down/768 kbps up). The vertical lines indicate the number of tokens for three different services when each user downloads tokens for all services in a city (city) or downloads tokens for all cities (all). (The JiWire and WiGLE results only examine the top 10 cities with the most hot spots.)

the bottleneck is a client on a typical cable modem (6 Mbps down/768 kbps up). The vertical lines indicate the number of tokens for three different services when each user downloads tokens for all services in a city (city) or downloads tokens for all cities (all). jiwire and wigle refer to the hot spot directories described earlier in this section. yelp refers to all physical establishments in Yelp.com's review database, which includes restaurants, grocery stores, bars, clubs, etc.[12] As in the previous section, we assume that the users of the service are distributed proportionally to each city's population. (The JiWire and WiGLE results only examine the top 10 cities with the most hot spots.)

This analysis shows that a service similar to Yelp.com could practically use our reporting protocol to protect the privacy of its users if these users were willing to reveal the city where they submit reviews (but not the actual establishments). The average user would download 10,449 tokens, which implies a per-user server cost of 0.2 cents and a download time of 7 seconds. If users were not willing to reveal the city where they submit reviews, then they would need to download tokens for all establishments, incurring a cost of 18 cents per user and a download time of 9 minutes. This cost may still be reasonable, but would be harder to justify for free or ad-supported services. In addition, this download time is qualitatively longer on human timescales (i.e., is likely viewed as a "download" rather than a click-through), which may deter users from joining.

Not all recommendation systems can subdivide their items by location (e.g., reviews for virtual product catalogs). Therefore, an important research 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.

6.4.2 Personalized Collaborative Filtering

In contrast to plain-query-based LBSes and crowd-sourced LBSes, personalized 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 CF techniques. For example, techniques to prevent Sybil attacks and fraud in such systems, such as DSybil [44], 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 multiparty voting [19], [20], these techniques are difficult to apply when clients are not always online. Thus, designing appropriate privacy models and mechanisms for CF services is an important area of future work.

7 RELATED WORK

Wifi-Reports is related to five areas of previous work: AP selection, e-cash and secure voting, recommender systems, personalized collaborative filtering, and collaborative sensing.

12. We calculate the number of establishments in each city by summing the number of establishments reported in each category on http://www.yelp.com.

AP selection. Salem et al. [38] also propose a reputation-based protocol for AP selection. In contrast to Wifi-Reports, their protocol requires changes to the standard 802.11 protocol, it does not protect clients' location privacy, it assumes APs can predict their performance, and it does not address varying wireless channel conditions. In addition, unlike this paper, their work did not evaluate its feasibility on empirical data.

The authors in [33], [39], [41] 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 6 that leveraging historical information outperforms direct measurement [33] 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 hot spot database. Nonetheless, our work is complementary to [39] and [41], which can better estimate the quality of the wireless channel when it is the performance bottleneck.

802.11k [13] is an industry standard now in development that will enable APs to give Wi-Fi clients performance information other than signal strength for improved AP selection. However, 802.11k is only designed to support the selection of APs within the same administrative domain (e.g., all the APs in a campus network). Wifi-Reports is designed to help users choose APs across different domains.

Electronic cash and secure voting. Wifi-Reports uses blind signatures in a manner similar to well-known electronic cash [21], [22] (e-cash) and secure voting [26] (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.

Recommendation systems. Having users report on items or services to ascertain their value is a well-known idea [15]. Wifi-Reports shares the most similarities with Broadband reports [2], which rates ISPs using user-reported speed tests (e.g., [8]) that measure their backhaul capacities. Unlike Wifi-Reports, 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. Furthermore, it is easier to limit Sybil attacks because a user is identified by an IP address, which is hard to spoof while maintaining a TCP connection. Finally, in contrast to APs, broadband measurements generally do not depend on the user's location.

Personalized collaborative filtering. Some recommendation systems use personalized collaborative filtering (e.g., [42], [44]) to mitigate the impact of 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 personalized CF techniques can limit the exposure of this information by using secure multiparty voting [19], [20]. 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., [31], [14]), but they do not address the unique challenges of AP measurement and reporting. Anonysense [23] is one such platform that uses group signatures to preserve location privacy and a trusted computing base (TCB) to mitigate fraud. Each of these mechanisms has limitations. First, group signatures ensure anonymity only if the authority is trusted not to reveal user locations. If it is not trusted, then it can deanonymize any report signature using its master key. By using blind signatures, Wifi-Reports does not need to make this assumption. Second, while a TCB can thwart software-based tampering, it cannot prevent hardware-based tampering (e.g., disconnecting a radio antenna). Nor can it prevent collusion between adversarial clients and APs. Therefore, a TCB alone is insufficient to prevent Sybil attacks; an account authority that limits the influence of any one user is necessary. Nonetheless, the Wifi-Reports measurement client could also leverage a TCB to mitigate fraud even more. For example, a TCB could be used to ensure that even malicious clients cannot make up reports without performing any measurements, raising the amount of effort required to generate a fraudulent report. A TCB could also be used to identify distinct devices securely to the account authority (e.g., using a hardware fingerprint), obviating the need for a credit card or other "real-life" credential to prevent large-scale Sybil attacks.

8 CONCLUSION

In this paper, we presented the first measurement study of commercial APs and showed that there is substantial diversity in the performance. Hence, selecting the best AP is not obvious from observable metrics. We presented Wifi-Reports, a service that improves AP selection by leveraging historical information about APs contributed by users. Wifi-Reports can handle reports submitted at different locations, protects users' location privacy, and is resilient to a small fraction of fraudulent reports.

Additional data from our measurement study and the components of Wifi-Reports that we built are available at CRAWDAD [34], [35].

REFERENCES

[1]   Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2, 2010.

[2]   Broadband Reports, http://www.dslreports.com, 2010.

[3]   HTTP Analyzer, http://www.ieinspector.com/httpanalyzer, 2010.

[4]   Illegal Sale of Phone Records, http://epic.org/privacy/iei, 2010.

[5]   iPass, http://www.ipass.com, 2010.

[6]   Jiwire, http://www.jiwire.com, 2010.

[7]   Skyhook Wireless, http://www.skyhookwireless.com, 2010.

[8]   Speedtest.net, http://www.speedtest.net, 2010.

[9]   T-Mobile Germany Blocks iphone Skype over 3g and Wifi, http:// jkontherun.com/2009/04/06/t-mobile-germany-blocks-iphone-skype-over-3g-too, 2010.

[10]   Wireless Geographic Logging Engine, http://www.wigle.net, 2010.

[11]   Fraud and Related Activity in Connection with Access Devices, Homeland Security Act (18 U.S.C. Section 1029), 2002.

[12]   "Wireless Location Tracking Draws Privacy Questions," CNET, http://news.com.com/Wireless+location+tracking+draws+ privacy+questions/2100-1028_3-6072992.html, May 2006.

[13]   *IEEE Std 802.11k-2008 (Amendment to IEEE Std 802.11-2007), IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (Phy) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs,* IEEE, June 2008.

[14]   T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich, "Mobiscopes for Human Spaces," *IEEE Pervasive Computing,* vol. 6, no. 2, pp. 20-29, Apr. 2007.

[15]   G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 6, pp. 734-749, June 2005.

[16]   L. Balzano and R. Nowak, "Blind Calibration of Sensor Networks," *Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN),* 2007.

[17]   M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme," *J. Cryptology,* vol. 16, no. 3, pp. 185-215, 2003.

[18]   A.R. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Pervasive Computing,* vol. 2, no. 1, pp. 46-55, Jan.-Mar. 2003.

[19]   J. Canny, "Collaborative Filtering with Privacy," *Proc. IEEE Symp. Security and Privacy,* 2002.

[20]   J. Canny, "Collaborative Filtering with Privacy via Factor Analysis," *Proc. SIGIR,* 2002.

[21]   D. Chaum, "Blind Signatures for Untraceable Payments," *Proc. Advances in Cryptology,* pp. 199-203, 1982.

[22]   D. Chaum, A. Fiat, and M. Naor, "Untraceable Electronic Cash," *Proc. Ann. Int'l Cryptology Conf. (CRYPTO),* pp. 319-327, 1990.

[23]   C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "Anonysense: Privacy-Aware People-Centric Sensing," *Proc. MobiSys,* pp. 211-224, 2008.

[24]   R. Dingledine, N. Mathewson, and P. Syverson, "TOR: The Second-Generation Onion Router," *Proc. USENIX Security Symp.,* 2004.

[25]   C. Doctorow, "Why Hotel WiFi Sucks," http://www.boingboing. net/2005/10/12/why-hotel-wifi-sucks.html, Oct. 2005.

[26]   A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Elections," *Proc. Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT),* 1993.

[27]   B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall, "Improving Wireless Privacy with an Identifier-Free Link Layer Protocol," *Proc. MobiSys,* 2008.

[28]   M. Gruteser and B. Hoh, "On the Anonymity of Periodic Location Samples," *Proc. Int'l Conf. Security in Pervasive Computing,* 2005.

[29]   D. Han, A. Agarwala, D.G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Mark-and-Sweep: Getting the "Inside" Scoop on Neighborhood Networks," *Proc. Internet Measurement Conf. (IMC),* 2008.

[30]   G. Judd and P. Steenkiste, "Using Emulation to Understand and Improve Wireless Networks and Applications," *Proc. Symp. Networked Systems Design and Implementation (NSDI),* 2005.

[31] A. Krause, E. Horvitz, A. Kansal, and F. Zhao, "Toward Community Sensing," *Proc. Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2008.

[32] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit, "Place Lab: Device Positioning Using Radio Beacons in the Wild," *Proc. Int'l Conf. Pervasive,* 2005.

[33] A.J. Nicholson, Y. Chawathe, M.Y. Chen, B.D. Noble, and D. Wetherall, "Improved Access Point Selection," *Proc. MobiSys,* 2006.

[34] J. Pang, CRAWDAD Data Set cmu/hotspot (v. 2009-04-15), http://crawdad.cs.dartmouth.edu/cmu/hotspot, Apr. 2009.

[35] J. Pang, B. Greenstein, and M. Kaminsky, CRAWDAD Tool Tools/ Collect/802.11/Wifi-Scanner (v. 2009-04-15), http://crawdad.cs. dartmouth.edu/tools/collect/802.11/Wifi-Scanner, Apr. 2009.

[36] J. Pang, B. Greenstein, D. Mccoy, S. Seshan, and D. Wetherall, "Tryst: The Case for Confidential Service Discovery," *Proc. Workshop Hot Topics in Networks (HotNets),* 2007.

[37] R.S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," *IEEE Network,* vol. 17, no. 6, pp. 27-35, Nov./Dec. 2003.

[38] N.B. Salem, J.-P. Hubaux, and M. Jakobsson, "Reputation-Based Wi-Fi Deployment Protocols and Security Analysis," *Proc. Int'l Workshop Wireless Mobile Applications and Services on WLAN Hotspots (WMASH),* 2004.

[39] K. Sundaresan and K. Papagiannaki, "The Need for Cross-Layer Information in Access Point Selection Algorithms," *Proc. Internet Measurement Conf. (IMC),* 2006.

[40] United States Census Bureau, Table 1: Annual Estimates of the Population for Incorporated Places over 100,000, http:// www.census.gov/popest/cities/tables/SUB-EST2007-01.csv, 2007.

[41] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley, "Facilitating Access Point Selection in IEEE 802.11 Wireless Networks," *Proc. Internet Measurement Conf. (IMC),* 2005.

[42] K. Walsh and E.G. Sirer, "Experience with an Object Reputation System for Peer-to-Peer Filesharing," *Proc. Symp. Networked Systems Design and Implementation (NSDI),* 2006.

[43] H. Yu, P.B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks," *Proc. IEEE Int'l Symp. Security and Privacy,* 2008.

[44] H. Yu, C. Shi, M. Kaminsky, P.B. Gibbons, and F. Xiao, "DSybil: Optimal Sybil-Resistance for Recommendation Systems," *Proc. IEEE Int'l Symp. Security and Privacy,* 2009.

Jeffrey Pang received the BS degree in computer science from the University of California, Berkeley in 2003, and the PhD degree in computer science from Carnegie Mellon University in 2009. He is a research scientist at AT&T Labs—Research. He is interested in distributed systems, mobile computing, wireless networks, privacy, and security. More details about his research can be found at http:// www.cs.cmu.edu/~jeffpang.

Ben Greenstein received the bachelor's degree in history from the University of Pennsylvania, and the doctorate degree in computer science from the University of California, Los Angeles, where he studied software systems for wireless embedded sensing devices with Deborah Estrin and Eddie Kohler. He is a research scientist at Intel Labs Seattle. He is interested in developing wireless, distributed, and potentially ubiquitous systems that improve everyday life.

Michael Kaminsky received the BS degree in electrical engineering and computer science from the University of California, Berkeley, in Spring 1998, and the PhD degree in computer science from the Massachusetts Institute of Technology. He is a senior research scientist at Intel Labs Pittsburgh and adjunct faculty in the Computer Science Department at Carnegie Mellon University. He joined Intel in Summer 2004. He is generally interested in computer science systems research, including distributed systems, networking, operating systems, and network/systems security. More details about his research can be found at http://www.pittsburgh. intel-research.net/people/kaminsky.

Damon McCoy received the PhD degree in computer science from the University of Colorado. He is a CRA-NSF computing innovation fellow in the Department of Computer Science and Engineering, University of California, San Diego. His research interests include improving the security and privacy of current technologies including automotive components and wireless devices.

Srinivasan Seshan received the PhD degree from the Computer Science Department, University of California, Berkeley, in 1995. He is currently an associate professor in the Computer Science Department, Carnegie Mellon University. From 1995 to 2000, he was a research staff member at IBM's T.J. Watson Research Center. His primary interests are in the broad areas of network protocols and distributed network applications. More details about his research can be found at http://www.cs.cmu.edu/~srini.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.