Social Wi-Fi: Hotspot Sharing with Online Friends

Zhen Cao, Jürgen Fitschen, and Panagiotis Papadimitriou

Institute of Communications Technology, Leibniz Universität Hannover, Germany
{zhen.cao,panagiotis.papadimitriou}@ikt.uni-hannover.de, mail@jfitschen.de

Abstract—Security and liability issues set barriers to WiFi sharing, giving away opportunities to capitalize the unused capacity in WiFi networks and provide wider Internet access. Considering the lack of trust between WiFi sharers and potential guests, we leverage on the increasing penetration of online social networks to enable WiFi sharing with online friends.

To this end, we present a WiFi sharing architecture, called Social WiFi, and the associated mechanisms for network discovery and authentication. Social WiFi couples Bloom Filter with extensions to existing authentication mechanisms for the discovery and authentication of guests to WiFi networks owned by their online friends. We assess the authentication in Social WiFi in terms of performance and efficiency. Furthermore, we discuss deployment issues and lay out a delegation framework for the migration of the existing WiFi network infrastructure to Social WiFi.

I. INTRODUCTION

WiFi is gaining increased popularity, especially for mobile data offloading [1] and Internet access sharing. Specifically, several initiatives (e.g., FON [2], PAWS [3], Virtual Public Networks [4]) enable the sharing of home broadband connections with the public, taking advantage of the density of wireless access points in residential areas. Internet access sharing may be either offered for free (PAWS, Virtual Public Networks) or can be available only to subscribers (FON).

Accessing Internet through shared home broadband connections can raise security and liability issues both for the sharer and the guest [5]. For example, a sharer will be concerned whether a guest is legitimate, since the sharer may be held accountable for any malicious actions of guests (e.g., Denial-of-Service attacks, illegal content download). Conversely, guests can become victims of malicious sharers that advertise fake SSIDs in their attempt to eavesdrop guests’ traffic and retrieve private information transmitted over the shared WiFi. Furthermore, a malicious sharer can assign a phony DNS server to the guest via DHCP, redirecting requests to bogus servers for phishing. These issues can put off users from sharing their Internet connection and potential guests from establishing connections to shared WiFi of unidentified owners. Essentially, most WiFi owners “lock” their hotspots by setting up shared secrets, giving away the opportunity to share with others.

The ever-growing penetration of online social networks (OSN) creates opportunities for Internet access sharing with friends. For example, consider that Alice is in the vicinity of Fred, Linda and George’s hotspots, which are Alice’s Facebook, LinkedIn and Google+ friend, respectively (Fig. 1).

Alice can hop on any of the shared WiFi of Fred, Linda or George, if there is a way to validate their social relationship. Such a community WiFi sharing motivates and incentivizes user participation, since it is essentially reciprocal. However, currently the WiFi network architecture lacks such a secure community sharing mechanism.

Based on these observations, we propose Social WiFi, an architecture that enables WiFi owners to share their network with their OSN friends. Essentially, Social WiFi allows a user to discover a WiFi network operated by one of his friends and authenticate to the network by validating their social relationship. Fig. 2 illustrates an overview of the proposed mechanisms. Bob owns a WiFi and advertises his friends list. When Alice comes to the vicinity of Bob’s hotspot, she discovers that she is a friend of Bob through the advertised information. Social WiFi discovery is followed by the mutual authentication between Alice and Bob, at which their friendship is validated through the information acquired via the social network API.

Social WiFi raises the need for network discovery and authentication based on social relationships. Currently, the SSID based network discovery and existing authentication methods do not fulfill these requirements. More precisely, SSID based WiFi discovery has turned out to be insecure, while the 32-byte length of SSID is insufficient for the encapsulation of social information, given the large number of friends associated with most social profiles. In addition, existing authentication methods mandate preshared secrets which are not feasible to distribute among a large number of social network participants. To this end, we present the extension of the standard Access Network Query Protocol...
(ANQP) to support social WiFi discovery, so that a guest can easily identify any hotspots operated by his friends. We further propose a new Extensible Authentication Protocol (EAP) method, namely EAP-Social, for mutual authentication between friends. EAP-Social can establish mutual trust by validating online friendship, obviating the need for shared key distribution beforehand.

The remainder of this paper is organized as follows. In Section II, we elaborate on the challenges faced by Social WiFi. Section III lays out the proposed architecture and enabling technologies. In Section IV, we present the evaluation results of Social WiFi, followed by its deployment considerations in Section V. Finally, Section VI provides an overview of related work, while Section VII highlights our conclusions.

II. CHALLENGES

Social WiFi entails significant challenges in terms of network discovery and authentication:

**Network discovery.** Social WiFi requires an efficient way for users to discover the hotspots operated by their online friends. The SSID based approach has several limitations. First, SSIDs can be easily spoofed and are, hence, not secure. There are several reports on fake SSID attacks [6], [7]. Second, the 32-byte length limitation of SSID prevents the network to encode adequate social relationship information for discovery. Furthermore, social relationship information is deemed as private and therefore, it should not be disclosed. To address these issues, we propose an extension of the standard access network query protocol and use Bloom Filter [8] to compress the social relationship information. Further details of our social WiFi discovery mechanism are given in Section III-A.

**Authentication.** EAP authentication is used in most WiFi networks [9]. Existing EAP authentication methods require a preshared secret or a preconfigured certificate in advance. Employing such techniques into Social WiFi would require the distribution of credentials among a large number of social network participants. Social WiFi obviates such need by enabling the mutual authentication between the guest and sharer. This is achieved by a validation of their friendship through a challenge-response procedure. The proposed authentication mechanism is discussed in detail in Section III-B.

III. SOCIAL WiFi

In the following, we discuss the main functions pertaining to Social WiFi.

A. Network Discovery

In order to enable users to discover the WiFi hotspots operated by their OSN friends, we leverage on a new feature provided by the most advanced IEEE 802.11-2012 standard [10], namely Access Network Query Protocol (ANQP). ANQP aims at providing carrier-grade WiFi services with a better network discovery capability and is supported by many mobile device manufacturers and network equipment vendors [11]. Essentially, ANQP enables the guest device to query specific information about the access network before associating to it. This feature is very useful and can be extended to enable guests to query their social relationship with the hotspot owner.

Fig. 3 illustrates the network discovery procedure. When a guest device comes to the vicinity of the hotspot, it receives beacons from the network announcing its support of ANQP (msg 1). After probing the network about its liveness (msg 2, 3), the device sends an ANQP query request message to the network (msg 4). This request contains the list of OSNs the user would like to query, such as Facebook, Google+, and LinkedIn. The access point (AP) subsequently retrieves the friend list information from the specific OSNs by revoking the corresponding APIs (msg 5). After acquisition of the friend lists, the AP sends them to the guest device in msg 6. If the guest discovers that he belongs to the friend lists, he associates and authenticates to the network (msg 7).

[^1]: We include msg 5 in the sequence for illustrative purpose, as this message can be pre-fetched obviating its transmission for each query.
In the ANQP query response of msg 6, the hotspot owner includes the friend list of each queried OSN, e.g., Facebook, Google+ or LinkedIn, using the APIs provided by these OSNs [12]. However, two issues arise, i.e., privacy and data compression. More precisely, disclosing friend names violates privacy. Furthermore, the encapsulation of all friend names into a single message is not feasible, especially when the hotspot owner’s online friends are in the range of hundreds or thousands [13]. To rectify this, we employ Bloom Filters for data compression and privacy protection.

Bloom filter (BF) [8] is a widely used data structure to compress an arbitrary data set into a bit vector and provide membership lookup. We leverage on BF to compress the friend list into a bit vector and allow a user to perform lookups in the list. Fig. 4 illustrates the compression and use of BF:

1) **BF initialization.** For a BF with a bit vector size of $K$, each bit is set to zero.

2) **BF construction.** For each name in the friend list, the BF computes $m$ hash functions $H_1(),...,H_m()$ of the name string (Fig. 4 shows only one hash function): $i_v = H_v(name) \mod K; v = 1, ..., m$. Then all the $i_v^{th}$ bits of the BF are set, $BF[i_v] = 1; v = 1, ..., m$.

3) **BF-assisted lookup.** For each queried name, e.g., Alice, the BF checks $BF[i = H_v('Alice') \mod K], v = 1, ..., m$. Alice belongs to the friend list only if all the $m$ bits are equal to 1.

The friend lists are initially compressed into BFs, and subsequently, the AP inserts the BFs into msg 6 in Fig. 3. The guest device then performs a BF lookup to check whether the guest’s name is in the friend lists. Only if there is a match, the guest device will attempt to associate and authenticate to that network.

BF brings three significant benefits to network discovery: (i) it preserves privacy because it does not disclose friend names, (ii) it achieves high compression of friend lists, since the BF occupies $K$ bits irrespective of the friend list size, and (iii) lookup performance does not degrade with the increase of the friend list size.

However, BF raises two issues, i.e., deletion of a friend from the list and false positives. These issues can be addressed as follows. A friend name can be removed using the counting BF (CBF) [14], which stores a counter value rather than a bit in each slot. The counter is decreased when a friend is removed. As long as the counter is greater than zero, CBF will report a match.

On the other hand, false positives can lead to wrong lookup results. For example, Fig. 4 depicts Alice and Fred hashing to the same value. Although only Alice is actually included in the friend list, the BF will report a match for both of them. Nevertheless, false positives do not cause any serious implication: they initiate authentication which will be unsuccessful, since the OSN relationship between the guest and the hotspot owner will not be validated (see Section III-B).

### B. Authentication

Existing authentication methods require a preshared secret or a preconfigured certificate for authentication. Since an online friend does not have a preshared secret, we cannot rely on existing methods for mutual authentication. Instead, we propose a new authentication method called **EAP-Social**.

Before elaborating on **EAP-Social**, we introduce the standard Extensible Authentication Protocol (EAP) [9]. EAP is the standard authentication framework adopted by most WiFi vendors and service providers. Fig. 5 illustrates a typical EAP authentication, among a client, an AP, and an Authentication, Authorization and Accounting (AAA) server [15]. After the client device attempts to connect to the AP, the AP initiates authentication by sending an EAP Identity Request (msg 1). The client responds with the authentication identity (msg 2). The AP encapsulates the EAP message into Radius and forwards it to the corresponding AAA server (msg 3). Subsequently, the AAA server selects a specific EAP method (e.g., EAP-TTLS/EAP-PEAP) and initiates the mutual authentication (msg 4–8). After a successful authentication, an EAP-
EAP-Identity Request
1. EAP-Identity Response (Alice’s Facebook name)

2. Call Facebook API

[Challenge Question for Alice: How many mutual friends does
Alice have with Bob?]

3. Response: hash of #of mutual friends

4. Client response: hash of #of mutual friends

5. AAA Server

6. EAP success

Alice Bob’s Hotspot AAA Server

Fig. 6. EAP-Social: a mutual authentication method based on validation of friendship.

Success message sent (msg 9).

During mutual authentication, the client and server prove
their ownership of a certain certificate or knowledge of a
pre-shared secret. For example, in EAP-TTLS [16], the server
uses a certificate, and client uses a password. Since there are
no such credentials in Social WiFi, we propose the EAP-Social
authentication method, at which both parties utilize
information from OSNs to establish trust.

Fig. 6 illustrates the sequence of messages exchanged
in EAP-Social. We assume Alice and Bob are friends on
Facebook. Alice discovers that one of the nearby hotspots
is operated by Bob using Social WiFi discovery (Section
III-A). Subsequently, Alice initiates the authentication to this
network. After receiving the EAP-Identity request (msg 1),
Alice responds with her Facebook user name (msg 2). The
AAA server in Bob’s network, in turn, examines whether Alice
belongs to the friend list. If so, the AAA server invokes the
Facebook API to generate a friendship challenge question for
Alice, e.g., the number of mutual friends (msg 4). In addition,
the AAA includes a number i and the hash value of the
i-th mutual friend’s name between Alice and Bob (the value
of i could vary for protection against spoofing). As such,
Bob demonstrates his knowledge about the answer without
disclosing it. Alice then validates Bob’s answer by invoking
the Facebook API to get their mutual friend list and repeating
the hash computation². If the answer is correct, Alice considers
the network as trusted and, subsequently, provides an answer to
the challenge question in msg 5. Similarly, the AAA server
validates Alice’s answer by comparing it with the expected
one. If it is correct, the authentication is successful and Alice
is granted Internet access.

We further discuss the security of EAP-Social. Since the
APIs provided by social networks are strictly authorized [12],
it is impossible for any malicious user to get the friend names
of third parties. In msg 4 (Fig. 6), if Bob can successfully
get the hash value of the i-th mutual friend, his authenticity is
certified. Similarly, if Alice is not a real friend of Bob, she

²Alice can either fetch this information from her cellular network or may
alternatively use friend list cached in the client device.

will not be in position to know the exact number of mutual
friends. Furthermore, since the answers of Alice and Bob are
hashed, eavesdropping is not possible.

IV. EVALUATION

Since the BF lookup efficiency have been validated in
previous work [8], [17], we hereby focus on the evaluation
of the proposed EAP-Social authentication protocol. To this
end, we have implemented the protocol in Mininet [18], [19].

The EAP-Social client has been implemented and integrated
into the Linux wpa_supplicant [20], a widely used open-source
WiFi authentication supplicant, while the EAP-Social authentica-
tion server has been implemented using the authenticator
software hostapd [21]. For the friend list information required
for a successful authentication, we assume both the client
and server have fetched them in advance and stored them locally
in configuration files. Since both wpa_supplicant and hostapd
have a full EAP authentication stack and support various stan-
dard EAP methods (e.g., EAP-TTLS/PEAP), we examine the
message exchanges of EAP-Social rather than the lower-layer
interactions (e.g., fragmentation and retransmission) during
authentication.

Using Mininet, we emulate a WiFi topology including an
AP and two hosts that are configured as the EAP-Social client
and the EAP-Social authentication server, respectively. The
wireless link between the client and the AP is configured
with a bandwidth of 2 Mbps, a loss rate of 1%, and an one-way latency ranging from 5ms to 30ms. We first measure the authentication delay, i.e., the time elapsed (on the server side) from the first EAP request till the final EAP-Success message. We further derive the authentication processing time by subtracting the propagation delay from the authentication delay. Eventually, we compare the proposed EAP-Social method with the standard EAP-TTLS implementation [16] (used in Eduroam [23]) in terms of authentication delay and performance.

Fig. 7 illustrates the authentication delay for EAP-Social and EAP-TTLS. According to the captured packet traces, EAP-Social completes the mutual authentication within two RTTs, as opposed to the six RTTs required by EAP-TTLS. This is corroborated by Fig. 7, where EAP-Social incurs lower authentication delay than EAP-TTLS.

Social WiFi yields higher efficiency than wireless sharing initiatives (e.g., Fon [2], Facebook WiFi [22]) that rely on captive portal authentication. More precisely, EAP-Social completes the authentication within 100 ms (Fig. 7), while there is no need for user intervention due to the built-in EAP auto-authentication capability on the guest device. In contrast, captive portal authentication requires that the user launches his web browser and enters an arbitrary URL, before he is directed to the portal page to enter his account information. This amounts to a total of more than ten clicks and at least 10 seconds of authentication delay [25].

Furthermore, Fig. 8 illustrates the authentication processing time for both protocols. EAP-Social yields a shorter processing time than EAP-TTLS, since the latter employs the most advanced public key cryptography (RSA or ECC) with signature generating and verification overhead. Instead, EAP-Social uses hash computations and a simpler authentication logic. Along these lines, EAP-Social comprises an efficient method for mutual authentication without any implications on user experience.

V. DEPLOYMENT CONSIDERATIONS

Hereby, we discuss some deployment considerations for Social WiFi.

A. Bloom Filter Configuration

Since Bloom Filter generates false positives, we investigate how to control the false positive rate at an acceptable range. The false positive rate $\alpha$ can be calculated using the formula [8]:

$$\alpha = \left(1 - \left(1 - \frac{1}{K}\right)^N\right)^{n_h}$$

(1)

where $K$ is the bit vector length, $N$ is the number of friends in the BF and $n_h$ is the number of used hash functions. The false positive rate decreases with the increase of $K$ and $n_h$.

Based on the distribution of number of friends in Facebook social graph reported in [13], 95% percent of users have less than 1000 friends. Along these lines, we consider $N \leq 1000$. The evolution of false positive rate with the number of friends is illustrated in Fig. 9. The length of BF is set to 2000 and 3000 bits respectively, for each of which we measure the false positive rate with $n_h$ set to 2 or 3.

With false positives, a user will be misled into connections to hotspots that are not owned by his friends. We can avoid this implication by controlling the false positive rate. For example, according to Fig. 9, using a bit vector longer than 3000 bits and at least three hash functions (i.e., $K \geq 3000, n_h \geq 3$) generates a false positive rate lower than 5%.

B. Migration to Social WiFi

Social WiFi requires a few modifications to the guest device and the WiFi access network infrastructure. For the guest device, a software update with the ability of social discovery and authentication is required. Given the availability of open source $\text{wpa_supplicant}$ software, the integration of Social WiFi mechanisms into Android, Linux and Windows devices (we have tested it on a Linux laptop) is straightforward. In order to facilitate the adoption and migration of the infrastructure, we propose the delegation of the social relationship discovery and authentication to a “Cloud AAA” (Fig. 10). In particular, the hotspot owners will register their APs to the Cloud AAA, and authorize the Cloud AAA to invoke their corresponding social network APIs. Subsequently, the Cloud AAA will fetch the friend lists and compress them into Bloom Filters for discovery. Only EAP-Social is required to be implemented on the Cloud AAA in order to handle the social authentication.

This approach brings significant benefits. First, the APs will be running in the pass-through model, obviating the need of substantial hardware or software updates. Furthermore, the centralized Cloud AAA is in charge of social authentication, which eliminates the need for a standalone AAA deployment at each hotspot. Essentially, our approach provides a migration path to Social WiFi.

VI. RELATED WORK

There are several initiatives for WiFi sharing. FON members share their home WiFi, and, in turn, get free access at Fon hotspots, with a footprint of more than 14 million hotspots worldwide [2]. FON requires the installation of a certain WiFi router and uses a special open SSID for network discovery.
This has been proved insecure, according to several reports of fake FON hotspots [7]. The authentication in FON redirects users to a web portal, which is prone to account hijacking through those fake hotspots.

Facebook WiFi [22] is a prominent example of WiFi sharing for business hotspots owners. This service incentivizes small enterprises to share their WiFi access by rewarding them with increased homepage credits. When users discover the Facebook WiFi, they are directed to the owner’s homepage and get free Internet access by logging in using their Facebook account. Similar to FON, Facebook WiFi supports only SSID discovery and portal authentication.

Eduroam [23] is a WiFi sharing project among global academic institutes. Eduroam WiFi supports EAP authentication and the wireless access in Eduroam is encrypted with advanced security standards. However, the Eduroam is based on static subscriptions and is only available to members of the academic community.

Public Access WiFi Service (PAWS) [3] seeks to enable free Internet access by sharing home broadband connections with the public. PAWS resorts to Virtual Private Network (VPN) for traffic isolation. However, VPN raises several issues, such as scalability limitations, increased power consumption in guest devices, as well as potential lack of experience and technical background from guests to setup a VPN (according to PAWS, guests may come from digitally deprived communities). Virtual Public Networks (VPuN) extend PAWS by outsourcing the management and control of crowd-shared home networks to third-party virtual network operators [4]. In particular, VPuN leverages on software-defined networking (SDN) to reduce the operational expenses of such crowd-shared networks and, therefore, incentivize hotspot owners to share their WiFi in a controlled manner.

Social SDN [24] enables hotspots owners to express network sharing policies based on online social relationships. These policies are then translated into traffic engineering decisions providing varying QoS levels to guests, depending on their OSN relationship with the sharer. Social SDN mainly targets at service differentiation, and, as such, it does not provide any mechanisms for network discovery and authentication.

Compared to these initiatives, Social WiFi leverages on OSNs to facilitate Internet access sharing with online friends. With the proposed secure network discovery and mutual authentication method, Social WiFi alleviates the security and liability issues both for the sharer and the guest.

VII. CONCLUSIONS

In this paper, we presented Social WiFi, an architecture to enable WiFi sharing with online friends. Social WiFi addresses two important issues in WiFi sharing, i.e., network discovery and authentication. We proposed an extension to the ANQP protocol allowing users to retrieve the friend list of the hotspot owner. We employed Bloom filter to compress the friend list into a short bit vector and preserve the privacy of OSN relationship. We further presented a new authentication method called EAP-Social, at which the sharer and guest establish mutual trust via the validation of their online friendship. Our evaluation results indicate the efficiency of EAP-Social. Finally, we discussed the deployment considerations for Social WiFi and laid out a delegation framework to facilitate its adoption and deployment.

REFERENCES