Global Grassroots WiFi Sharing

Wolfgang Kiess, Till Elsner, Björn Scheuermann, and Martin Mauve
Computer Networks Research Group, Heinrich Heine University, Düsseldorf, Germany
{kiess, elsner, scheuermann, mauve}@cs.uni-duesseldorf.de

***Abstract*— Access point sharing communities are an attractive solution for low-cost global mobile Internet access. However, existing communities all require a central authority in form of a certificate authority or an authentication server. In this paper, we show how such a community network can be created without a central entity. We introduce a mechanism called remote station approval; a host can use it to offer Internet access without requiring a central instance and without the risk of legal responsibility for the guests' traffic. We discuss how our system preserves the anonymity of its users and present a prototype implementation and some first performance figures.**

I. Introduction

The availability of broadband Internet connections in combination with flatrate pricing models and cheap WiFi hardware has made sharing this connectivity via WiFi an attractive solution for mobile Internet access. This approach has the potential to provide high bandwidths at low or even no deployment and usage costs. Access point (AP) sharing communities make use of this fact: a mobile user (called *guest*) may use the access point of another user (the *host*) at its current location to establish a connection to the Internet.

Existing access point sharing approaches [8], [12], [18], [21] are built around a central authority, the *community operator*. The operator manages user accounts, issues certificates, and is generally in charge of community membership. Consequently, all existing approaches require a host to trust the community operator when it comes to verifying the permission of guests to access the Internet. Because a guest could generate malicious traffic which would appear to originate from the host, making Internet access available to other people may incur certain risks for the host [18]. As a consequence, the commitment to trust the community operator when it comes to granting Internet access is problematic as it can have legal consequences. This may deter potential hosts from participating in the community. We therefore look at a system where the host himself can make an autonomous, safe decision whether access should be granted or not. Consequently, such a system does not *need* a central managing entity: it is a *loose peer-to-peer community* without formal membership, formed solely by de-facto participation and adherence to a set of open protocols.

Obviously, such an approach contrasts with the quid pro quo principle of existing WiFi sharing systems which requires each member to provide Internet access to others in exchange for being allowed to use other APs. However, such a regulation is virtually impossible to enforce also in a centralized system. How should any possible mechanism decide whether an access point is intentionally unavailable or if the operating user's Internet access is located in an electromagnetically shielded location? How to distinguish between a host who does not service any guests intentionally and one who lives in a rural region where simply nobody demands its service? But despite the lack of enforcement, a lot of access points *are* made available [9]. We therefore conjecture that many potential hosts *are* willing to altruistically share their Internet access even if this were *not* required to use other APs. Furthermore, we even suspect that there might be many more open APs if currently existing negative incentives—first and foremost the need to trust the operator—can be overcome. Interestingly, a similar effect was observed in the design of the Tor anonymization overlay: by allowing each onion router to freely decide on its exit policy (defining addresses and ports reachable via the router), contributors can avoid legal and organizational risks—resulting in a larger number of routers and better service [4].

The central question that therefore arises is one of trust and authentication. Can we build a system in which the host need not trust anyone, and can still decide that forwarding an (unknown and untrusted) guest's traffic is safe? Here, we will introduce a design that overcomes this seemingly paradoxical situation, by a mechanism called *remote station approval*. The key idea is that a communication partner on the Internet (the *remote station*) can disburden the host from worries about potentially malicious guest traffic by explicitly agreeing to receive that traffic. Only after approval has been obtained, the guest is allowed to communicate with this one specific communication partner. This protects the mediating host from being made responsible for the guest's actions. The remote station will typically be a device in the guest's home network, in many cases it will be implemented in the guest's own access point. Then, the remote station approval concept can be combined with an idea from [18]: the remote station can relay the guest's Internet traffic, thereby making a fully decentralized, peer-to-peer AP sharing community possible.

The indirect routing of a guest users' traffic via its home access point can result in a performance degradation due to the introduced triangle routing. This has already been observed and discussed in previous work, and ways to alleviate this problem have been devised. Heer et al. [12], for instance, suggest to offer a corresponding relay service in the network core. However, regardless of the specific realization and location of the guest user's relay, the problem of authorizing the traffic to the relay remains. This is where the key contribution of this paper steps into the breach: our remote station approval mechanism allows to perform this authorization and the exchange of necessary cryptographic secrets in a completely distributed way and without any central authority.

It turns out that this approval mechanism can be designed in a way that exhibits a number of additional benefits. The mechanism we introduce does not require any communication between the guest and any other node on the Internet before approval from the remote station has been obtained. At the same time, it does not require the guest to disclose his identity. In typical usage scenarios, it is not even possible for the host to recognize a returning guest. All this is possible without asymmetric cryptography; this is a desirable feature because asymmetric cryptosystems require high computational efforts and are therefore prone to denial of service attacks.

In the remainder of this paper, Section II reviews related work and Section III presents our approach. Section IV then discusses a number of different attack scenarios before Section V presents a prototypical implementation and a performance evaluation. Section VI concludes the paper.

II. Related Work

The most well-known access point sharing community FON [8] is built around a central, commercially operated authority that verifies user identities. Users logging on to a FON host access point are authenticated via an authentication server, using a combination of username and password. FON exhibits the typical problems of a strongly centralized approach, for example with regard to the service availability: the system is only available if the central authentication server is reachable. The FON community operator can furthermore easily generate movement profiles of the guests, from the history of their authentication processes. These disadvantages could be partially overcome by the recently proposed AGE approach [21]. It uses centrally issued certificates and a distributed certificate revocation list for the authentication task. Nevertheless, it is still possible for the host to recognize a guest, which may allow to create movement profiles. Furthermore, as in FON, the guest's traffic is routed directly to the Internet by the host access point. This shifts the liability, i. e., the legal responsibility for this traffic, to the host. Finally, FON (and AGE) access points challenge the security of the users' traffic as it is transmitted potentially unencrypted over the wireless link and the host access point. None of these problems exist in the architecture that we are proposing.

A solution approach for the latter issue is offered by Sofanet [20], which provides VPN tunnel endpoints for rent. This allows hotspot users to route their IP traffic via an encrypted tunnel to the VPN provider and from there to the Internet, instead of sending it unencrypted over the wireless link and through the (untrusted) hotspot AP. This does, however, not address the host's liability problem (because guests may still send traffic directly to the Internet, using the host's IP address), and it does also not solve the movement profile issue.

To avoid not only the security problems of unencrypted traffic via foreign access points, but also the liability problems for the host, Sastry et al. [18] and Heer et al. [12] propose to establish a *mandatory* encrypted tunnel between the host AP and a trusted relay associated with the guest. The guest's traffic is then routed via this relay to the Internet. In the simplest case, the trusted relay is the guest's own access point (the *home* AP) or a Sofanet-like rented endpoint.

Sastry et. al.'s approach is targeted to local communities within one city. Fixed addresses from a private subnet are assigned to each user, limiting the system's scalability. Heer et al. build upon the Host Identity Protocol (HIP) [15] which they adapt for their system. This overcomes the scalability issues, but requires a modified network stack in all participating devices, which may constitute a severe practical drawback for user acceptance and deployment. Furthermore, neither Sastry et al. nor Heer et al. consider an AP sharing community without a central authority for user authentication (either as an authentication server or a certificate authority (CA)), resulting in the problems outlined above. We avoid such an authority and present a solution that can be implemented in user-space.

Authentication schemes in GSM [7], Kerberos [16], or LTPA [1] share the basic idea of a client A (e. g., a mobile phone) authenticating towards a service providing entity B (like a base transceive station) with the help of an authenticating entity C (for instance, a GSM authentication center), so parallels are apparent. However, the trust relationships in these systems differ in one central aspect: in these systems, there is trust between B and C, both of which are typically associated with the service provider. In our system, such a trust relationship does generally not exist. In some sense, we do therefore not authenticate A with the help of C, but A and C *conjointly* prove towards B that they "belong together".

III. Approach

The two key goals of this paper are to strengthen the host's position as an altruistic, yet fully autonomous entity while at the same time removing any central authority and authentication system. From a practical perspective these aims may appear contradictory: in existing systems, the central authority is the means to provide authentication. It is used to authorize and legitimate the guest. If this central authority is to be removed, how can the host be sure that it is (legally) safe to forward a guest's traffic?

Ideally, not one single packet generated by the guest should be forwarded unless the host knows for sure that this packet will not cause any harm. Of course, the host cannot definitely verify whether a guest generated packet is malicious, thus the intended *recipient* of the transmission should be asked. If the recipient explicitly agrees to receive the guest's traffic, it takes over the risk from the host access point. And just this is where we can tackle the problem: we replace the centralized or central-CA-supported authentication process by a fully decentralized and authority-free mechanism to obtain approval from the guest's intended communication partner. We call this communication partner the *remote station*. In an AP sharing scenario, it will be a trusted relay like the guest's home AP or a Sofanet-like tunnel endpoint.

In contrast to the host, the remote station does have a trust relationship with the guest. This trust relationship allows the guest to prove to the host that the remote station agrees to receive his traffic. In this section, we will show how this

can be realized in a completely decentralized fashion, without direct communication between guest and remote station before their trust relationship is verified by the host. We call this mechanism *remote station approval*.

A. Remote Station Approval

When arriving at a host access point, the guest first obtains a local IP address (typically via DHCP [5]). The guest will then inform the host access point about his intention to talk to his remote station. Because there is no central trust entity, it is not possible to tie the guest's identity in any way to a specific remote station; therefore, the guest could, in principle, name any arbitrary Internet host[1], claiming that it is "his" remote station. Therefore, further steps are necessary to convince the host access point that this guest can indeed be safely allowed to talk to the named remote station.

The scenario we are confronted with is thus a rather uncommon one: we have two parties—the guest and the remote station—that mutually trust each other, and whose aim is to prove this to a "man in the middle"—the host—, without being allowed to communicate directly. The key idea to solve this is to let the remote station create and transmit a secret to both host and guest in such a way that the host can verify that the claimed trust relationship exists.

The protocol handshake is outlined in Figure 1. When a guest G claims that he is allowed to talk to a specific remote station R and issues a *connection request*, the host access point H will contact R. This *key request* is tagged with a randomly generated nonce[2] n. As an expression of their trust relationship, G and R share a symmetric secret key s. This key is leveraged to prove the existence of the trust relationship to the host. Upon receiving H's request, R will generate a temporary, symmetric encryption key t (i. e., essentially a random number). This key is used only during this connection, as shared secret between host (!) and guest. t is then encrypted with the shared secret s, and the result $s(t)$ and t itself as well as the key request nonce n are returned to the host. While it might seem awkward to pass t to the host in clear text, note that the ability to intercept t already requires a situation where the possession of t does not provide any advantages. We discuss this in detail in Section IV.

H subsequently passes $s(t)$ on to G while keeping the unencrypted key t for himself. If the guest shares a trust relationship with the remote station, he knows the shared key s and can obtain t by decrypting $s(t)$. Thus, H and G are now in possession of the same shared secret t.

To verify this, packets from the guest to the remote station are authenticated on the hop between guest and host, using key t. The guest attaches a sequence number sn, used to prevent replay attacks, and a hashed message authentication code (HMAC) [13] to each of his packets p, proving that he is in possession of t and thus trusted by the remote station. Only valid packets are forwarded to the remote station. Note that this kind of packet authentication can easily be implemented without modifications in the guest's or host's network stack by setting up an authenticated tunnel between the guest and the host using key t. We can even employ existing standard VPN software for that purpose.

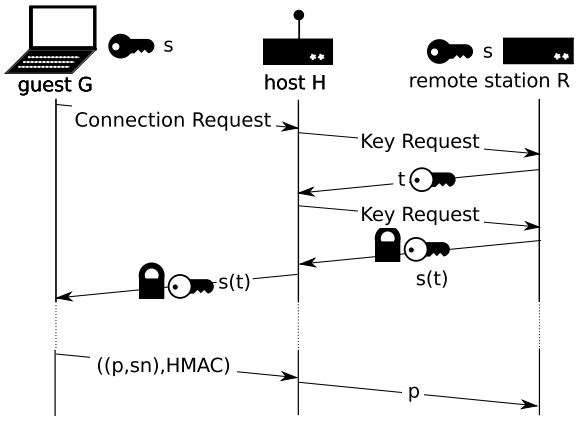

Fig. 1. Remote station approval.

If there is no trust relationship between G and R—i. e., the guest wrongly claimed that R is a trusted remote station—, then there are two possible cases. Either R does participate as a remote station in the community, but not for this guest, or R is some other Internet host. In the former case, R will understand H's request and will provide a pair $(t, s(t))$. But since G does not have the matching secret key s, he cannot recover t and cannot generate correct HMACs. In the latter case, R will probably not understand H's request to provide an authentication key pair; in any case, it will not provide a valid response to H's request. Consequently, H will not allow G to send packets to R.

B. Internet Connectivity for the Guest

Note that, from the host's perspective, the *nature* of the traffic between the guest and a consenting remote station does not matter at all. There is no fundamental difference whether Web or VPN traffic or any other protocol is used by the guest. Thus, from the starting point of a network with altruistic hosts willing to make Internet access available to strangers, we have arrived at an entirely new paradigm for deciding whether forwarding a guest's traffic is safe for the host. Of course, at this point, the guest will only be allowed to talk to the consenting remote station, but not to arbitrary Internet hosts. However, combining remote station approval with a mechanism that lets the guest establish an encrypted tunnel to the remote station [12], [18], using it as a relay as sketched in Figure 2, allows to provide full Internet access to the guest, still without liability problems for the host: from the perspective of any Internet host, a guest's traffic originates from its remote station. Unlike previous solutions, our system does not require centrally managed identities to achieve that.

[1]Note the double meaning of "host": the term *host* or *host access point* refers to the access point used by the guest to establish a connection, whereas an *Internet host* is an arbitrary computer connected to the Internet.

[2]A random number used once. As TCP can be used as transport protocol for the handshake between host and remote station, the initial TCP sequence number already serves the purpose of this nonce [2].

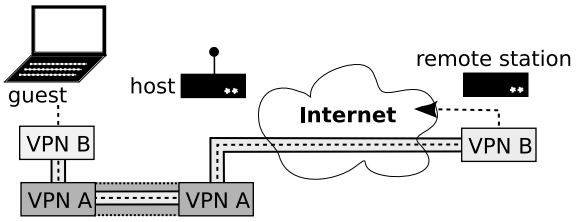

Fig. 2. VPN tunnel structure.

In the simplest case, the remote station will be the guest's own access point at home. Since this AP will typically have a dynamically changing IP address, a dynamic DNS service like DynDNS [6] can be used to locate such a remote station.

An apparent drawback of relaying all Internet traffic of the guest via the remote station is that it implies triangle routing, incurring, e. g., additional delays in the network. This, however, seems to be a reasonable price for disburdening the host from being held responsible for the traffic of its guests. Furthermore, our experimental results presented later on show that practical usability is virtually unaffected for most applications. (As a side note: triangle routing does not always decrease performance; in a different context it has been observed that it can potentially even *increase* performance [19].)

As pointed out in [12], a Sofanet-like approach with "rented" remote stations is an alternative to using the guest's own home network. This does not only eliminate the bandwidth limitation caused by the guest's own access point's upstream (which is assumed to be connected to the Internet by an asymmetric broadband connection), but also tackles the triangular routing problem in a simple, yet effective way: since those "rented" remote access point are expected to be available at different locations, the geographically closest remote station can be used, which reduces the triangular routing's performance impact to a minimum.

Note that the system is not in any way limited to one such remote station provider or to one specific dynamic DNS service. Each mobile user is entirely free in his decision to choose any provider on the market, or to stay fully independent by setting up his own service. In all these cases, he can use any host access point made available in the loose WiFi sharing community. Therefore, even though dynamic DNS services as well as commercially offered remote stations are "infrastructure services", neither of both constitutes a single central instance in the network, neither is vital for the network to operate, and neither requires that all network participants (especially not the hosts) trust them.

Privacy and data security concerns are overcome, because from the perspective of the host (and all other users of the host access point) all traffic by the guest is encrypted. The remote station serves as a *trusted relay* for the guest.

Moreover, many tunneling solutions that could be used by a guest also easily provide guest mobility without additional effort. For instance, OpenVPN [17], which we utilize in our prototype implementation discussed later on, typically uses IP addresses from a private subnet within the encrypted tunnel. In combination with NAT this allows to keep, for instance, TCP connections open when the IP address of the guest changes. Therefore, if the guest moves on to another host access point, he may simply move his existing tunnel to the new host access point (after a new remote station approval handshake), thereby keeping all his connections to Internet hosts open.

C. Guest Anonymity

A central problem in many existing access point sharing approaches is a lack of anonymity and privacy of the guest. In systems like FON [8], where authentication is performed in a centralized way, the community operator can track and locate user logins, allowing the operator to generate movement profiles. This is surely not desirable. Since the solution with an encrypted tunnel to a trusted relay solves the liability problem for the generated traffic by making it appear to originate from this relay, it is not necessary that the host learns the identity of his guests. Therefore, ideally, no entity in the AP sharing network except for the remote station should be able to infer the guest's identity.

Some existing systems, most notably [12], explicitly take anonymity into consideration. However, while they hide the real-world identity of the guest from the host, there is still a guest-specific ID in the guest's packets, making the guest recognizable. But this should be avoided, too: for instance, hosts might—alone or collaboratively with others—trace the guest's appearances and connection durations, analyze his behavior (working hours, movement profiles, ...), combine this with other information sources and may ultimately even reveal the real-world identity of the guest.

In contrast to existing systems, the remote station approval handshake does not reveal any persistent ID to the host access point and can thus be extended such that it provides what we call *unrecognizable anonymity*: the host is not able to recognize a guest as having visited his access point before. This notion is similar to *long-term unlinkability* defined—in a somewhat different context—in [10], but we achieve it in user-space. Achieving this level of anonymity is another central difference to authentication schemes like those in GSM [7] or Kerberos [16]: in GSM, the mobile station needs to be identified by a unique ID seen by the base transceive station; Kerberos could generally provide anonymous key material, but only if direct communication between client and authenticating server is allowed.

For the key material exchanged, the preservation of unrecognizable anonymity is immediately clear as a new session key t is generated upon each connection. However, a client that strives for full anonymity must also take care of other persistent attributes visible to the host. These include the MAC address of the guest's WiFi device and the address of the remote station. The MAC address issue is relatively easy to solve by randomly changing the WiFi adapter's MAC address before each connection [11]. The indirect identification via the used remote station is more tricky to address. If it is a Sofanet-like relay, many guests use the same relay, thus

obfuscating single guest identities. If the remote station is located at the guest's home and connected to the Internet via a common broadband connection like DSL, the remote station's IP address will change often enough so that it becomes inapplicable to recognize the guest over a longer time span. However, a regularly changing IP address makes need for a (dynamic) DNS name for the remote station to be reachable by the client, which again could identify the client.

To alleviate this issue, multiple DNS names could be registered per remote station, so that the guest can randomly pick one. However, the problem could also be tackled in a more fundamental way, by extending DNS with anonymization techniques. These could encrypt the host name part of the DNS request such that it changes with every request, so that only the authoritative name server is able to recover it. To this end, it is necessary to create an extension for DNS. The host name part *myname* of a domain name *myname.somedomain.org* is then not transmitted in cleartext, it is instead concatenated with a random string, encrypted, and ASCII encoded. Such a system, which would seamlessly integrate into existing DNS, is currently subject of ongoing research and not further described here due to space contraints.

IV. Attack Scenarios

In this section, we discuss a number of different attack scenarios that can be conceived in connection with the proposed access point sharing system. To this end, it is examined what can be achieved by attackers with different abilities.

It is very easy for an attacker to acquire access to the wireless connection between host and guest. For example, such an attacker can try to overtake the guest's current identity in the host's wireless LAN to pretend that he is an already authenticated guest. This could easily happen if the attacker starts to use the guest's IP and MAC address in the very moment the guest disappears from the host (without correctly closing the connection, e. g. by leaving the host's radio range), or the attacker may even be able to force the guest to disconnect from the host. Furthermore, an attacker with access to the wireless channel could also overhear the communication on the wireless medium or connect to an arbitrary host and request approval from an arbitrary remote station. In all these cases, the attacker is faced with the problem that he must correctly authenticate packets. Although the attacker may be able to overhear $s(t)$, this does not provide the corresponding session key t necessary for this task.

We can also think of situations where an attacker *can* successfully obtain t and convince a host to let his traffic pass through. To this end, the adversary—in addition to his wireless connection to the host—must be able to intercept traffic at some point along the Internet route between the host access point and the remote station. In fact, such a powerful attacker could even mount a man-in-the-middle attack, by claiming that any arbitrary machine X on the Internet is his remote station, waiting for the key request from the host to X, and then spoofing a reply with X's sender address and the correct nonce n from the intercepted key request. This would fool the host and allow the attacker to send arbitrary traffic to X via the host. However, if the attacker already does have the ability to intercept Internet traffic between the host and X, and can furthermore inject IP packets with a spoofed sender address, he does not really gain anything from such an attack: for such an attacker it would likewise be easy to attack X directly, potentially even spoofing the source address of the attack to make the packets look as if they originated from the host.

A different kind of attack could target the anonymity of the guest. Although we removed every persistent identifier to make the guest unrecognizable, its anonymity can still be compromised on a physical level as proposed in [3]. However, this weakness is caused by hardware characteristics and is thus not solvable in software. Beyond that the host does not pose any threat neither to guest nor remote station: they are connected via an encrypted VPN tunnel, and the only way the host could interfere with this connection is by interrupting it. A host can also not initiate a VPN connection to a foreign remote station, as he does not possess the required key.

V. Implementation & Performance

We have implemented a prototype software for an AP sharing community. It comprises the guest part, the host, and the remote station server (in the typical form to be installed on an access point).

In our implementation, the host's access point initially accepts associations from arriving guests and assigns them private IP addresses via DHCP. In this phase, the host's firewall blocks all traffic between the wireless network and the Internet. Guests may only issue connection requests to the host as described in Section III-A. If successful, guest and host have both obtained the same shared secret t. To connect to the remote station, the guest will then consecutively set up two tunnels A and B as outlined in Figure 2. First, an authenticated (but unencrypted) tunnel A is established to the host using key t, to allow the host to authenticate the guest's packets. This tunnel uses t to attach an HMAC to every packet sent, so the host can verify that the packets originate from a guest in possession of t. Upon establishment of this tunnel, the host adjusts its firewall rules to let packets traverse if they arrive via the guest-host tunnel A and are destined to the matching remote station or vice-versa. Only authenticated traffic will thus pass to the remote station. Through this first tunnel, the guest will typically set up another, encrypted and authenticated tunnel B to the remote station, which can then be used for accessing the Internet.

Both tunnels are implemented with OpenVPN [17], a platform-independent userspace VPN daemon based on TUN/TAP virtual network devices. All that is needed in terms of additional software for our AP sharing system are therefore user-space applications performing the remote station approval handshake; after a successful handshake, this software sets up the firewall rules at the host, fires up OpenVPN for the tunnels A and B, and provides all OpenVPN instances with the correct keys. This simplicity makes the system both easy to implement and robust.

TABLE I

LATENCY COMPARISON.

RTT in ms	Attempt 1	Attempt 2	Attempt 3
Guest to Host	0.3	0.3	0.3
Host to Remote Station	35.2	35.5	35.4
Remote Station to www.google.de	28.0	27.9	30.0
Guest via Host and Remote Station to www.google.de	65.9	73.9	77.3
Overhead	2.3	10.3	11.6

As the proposed scheme requires triangle routing via the remote station, we were interested in the performance degradation imposed by this indirection. To measure this, we have created the setup shown in Figure 2 in a city-wide deployment with this software. The host is located in our institute's laboratory and the remote station was installed at another location and was connected to the Internet via a private DSL link (6 Mbit/s downlink, 256 kbit/s uplink). In a first step, we examined unencrypted direct connections between the single entities. We concentrate on the round trip times (RTT) here. The results can be found in Table I in lines 2–4; they are averages over 100 samples each. The latencies are, as expected, very small for the local connection and a few tens of milliseconds over the wireline links. The line "Guest ... to www.google.de" then shows the RTT of the VPN-tunneled connection from guest via host and remote station to www.google.de, which is used in the role of an arbitrary Internet host. The "Overhead" line shows the difference between this tunneled connection and the sum of RTTs of the single hops, indicating that the overhead caused by routing and cryptographic operations in the triangular connection is not a bottleneck. As can be seen, routing traffic via a remote station deteriorates the service not further than to the expected extent due to the triangle routing: the latencies essentially add.

So, even if the guest's traffic is routed via a remote station located on another continent, we can expect this to increase the one-way latency by typically no more than 50-100 ms[3]. If we now consider the worst case of an intercontinental traveller who accesses a server right at her current location with triangle routing forth and back over the intercontinental link, the total increase in latency will not exceed 200 ms. Even in this exceptional situation, the performance would suffice for many applications—not only for Web and email, but also, for instance, for VoIP calls.

With respect to the bandwidth available to the guest, the impact of triangle routing is similar: it is constrained by the remote station's uplink. The header information of 69 bytes added by OpenVPN (for VPN, UDP, and IP headers) does not affect the bandwidth significantly and may even be compensated by OpenVPN's LZO compression. Already with the current asymmetric Internet connections, the available bandwidth will be adequate for the vast majority of applications. Note also that latency and bandwidth can be improved further with Sofanet-like remote stations closer to the current host.

[3] 50 ms: our own measurements of a Europe/US link; 100 ms: results reported in [14].

VI. CONCLUSIONS & FUTURE WORK

In this paper, we presented the first WiFi access point sharing approach that is not built around a central authority. This step is motivated by the insight that such a central entity must be trusted by all parties, which may deter potential participants. We showed how an access point that forwards traffic for other users can altruistically ask for approval from the user's desired communication partner. This is safe for the access point operator, without the need for any central instance for authentication. The approach also has a number of other desirable features, amongst them a much higher level of anonymity for the guest than in existing systems.

The next step is to evaluate the approach in a fully-fledged field test. Due to the sole usage of symmetric cryptography, already entry-level devices like the Linksys WRT54G router have sufficient computational capabilities to support the necessary operations, first measurements on this device show that the whole remote station approval handshake can be performed in 2 seconds (including network latencies). We will deploy a larger number of these devices in the city area to assess the performance. Furthermore, we are working on the DNS anonymization outlined in Section III-C and will soon provide an open source version of the above described software.

REFERENCES

[1] WebSphere and Domino single sign-on. http://www.ibm.com/developerworks/ibm/library/it-0101art2/.

[2] S. Bellovin. Defending Against Sequence Number Attacks. RFC 1948 (Informational), May 1996.

[3] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *MobiCom '08*, Sept. 2008.

[4] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security '04*, Aug. 2004.

[5] R. Droms. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard), Mar. 1997. Updated by RFCs 3396, 4361.

[6] DynDNS: Dynamic DNS. http://www.dyndns.com/services/dns/dyndns/.

[7] ETSI. GSM Recommendation 03.20. Technical Specification.

[8] FON. http://www.fon.com/.

[9] Map of available FON access points. http://maps.fon.com/.

[10] B. Greenstein, D. Mccoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *MobiSys '08*, pages 40–53, June 2008.

[11] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, 10(3):315–325, 2005.

[12] T. Heer, S. Götz, E. Weingärtner, and K. Wehrle. Secure Wi-Fi sharing on global scales. In *ICT '08*, June 2008.

[13] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), Feb. 1997.

[14] T. Leighton. Improving Performance on the Internet. *Communications of the ACM*, 52(2):44–51, Feb. 2009.

[15] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201 (Experimental), Apr. 2008.

[16] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021.

[17] OpenVPN. http://www.openvpn.net.

[18] N. Sastry, J. Crowcroft, and K. Sollins. Architecting citywide ubiquitous Wi-Fi access. In *HotNets '07*, Nov. 2007.

[19] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. *ACM SIGCOMM Computer Communication Review*, 29(4):289–299, 1999.

[20] Sofanet. http://www.sofanet.de/.

[21] N. A. Thompson, Z. Yin, H. Luo, P. Zerfos, and J. P. Singh. Extended abstract: Authentication on the edge – distributed authentication for a global open Wi-Fi network. In *MobiCom '07*, Sept. 2007.