Pervasive Content-centric Wireless Networking

Hanno Wirtz, Matteo Ceriotti, Benjamin Grap, Klaus Wehrle
Chair of Communication and Distributed Systems, RWTH Aachen
{wirtz, ceriotti, grap, wehrle}@comsys.rwth-aachen.de

***Abstract*—Current communication approaches assume network associations as a prerequisite to both content discovery and access. This *network-centric* paradigm incurs substantial communication and time overhead, as devices build knowledge of content availability only after blindly associating to a network. In pervasive mobile networking, this prerequisite and associated overhead prevents devices to efficiently identify desired communication partners, i.e., devices running an application or providing content of interest, within the broad mass of devices in communication range as found in everyday scenarios.**

SO-Fi, instead, realizes *content-centric* wireless networking by enabling pervasive content discovery before establishing a network infrastructure. SO-Fi builds on the IEEE 802.11 wireless broadcast medium to instantly achieve a discovery scope covering all devices in communication range. Realizing content discovery outside of secure network associations, SO-Fi supports use-case-specific communication security, i.e., confidentiality, WPA2 network security, DoS robustness, and user authentication. We show SO-Fi's feasibility and performance through real-world experimentation. Indeed, SO-Fi makes instant, content-centric wireless networking readily accessible to application designers.

I. Introduction

Offloading Internet application traffic and data to mobile wireless networks has been recognized as a viable solution to mitigate overloaded carrier networks. In this, opportunistic device-to-device networking [1]–[3], leveraging the proliferation of mobile devices, provides a networking structure to support offloaded application communication. Supporting this notion of utilizing device-to-device communication, recent results show the existence of a critical mass of contacts between devices in which communication may occur [4]. The gainful and timely utilization of contacts then determines [5] the success of the respective (offloaded) applications [6]–[8]. In such scenarios, given the number and diversity of devices and applications, the key challenge is to search for and connect to the *right devices* at the *right time*. In essence, this requires immediate and direct discovery of devices in communication range that participate in the same application, i.e., by providing or requesting *content of interest*.

As Fig. 1 illustrates, the *network-centric* design of 802.11 requires instantiation of and association to a single network prior to communication and subsequently restricts the discovery scope to the network, negating the, in principle, pervasive wireless discovery scope. This forces discovery to occur through either i) iterative association to observed networks and subsequent discovery, entailing enormous time and communication overhead, or ii) global provision of a designated ad-hoc network, which appears to be impractical. Outside of an existing network, device [9] and network discovery, e.g., via 802.11 scans, can not indicate content availability or application interests; conversely, content discovery mechanisms require a shared

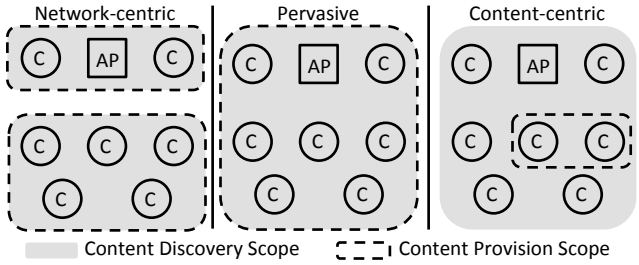

Fig. 1. Network-centric communication, as in 802.11, restricts content discovery and provision to the boundaries of disjunct networks, In contrast, pervasive communication envisions complete coverage of all reachable devices but would require maintaining a single permanent (ad-hoc) network infrastructure. SO-Fi enables content-centric communication, realizing a pervasive discovery scope and efficient, immediate provision of content between the requesting and providing device.

network. To overcome this deficit of semantics in wireless discovery, previous approaches [10]–[13] overload Bluetooth or 802.11 protocols to continuously *push* selected application information or rely on coordination by global higher-layer entities [14], [15].

In this paper, we propose the opposite, i.e., a *pull* mechanism that broadcasts content requests in standard 802.11 frames, exploiting the full pervasive scope of wireless communication for discovery. As shown in Fig. 1, discovery thus triggers instantiation of corresponding *content-centric* 802.11 networks only at devices that provide the requested application or content. Applications can thus "demand" a specific 802.11 network for purposeful communication from the surrounding devices. SO-Fi (Secure On-demand Wi-Fi) thereby i) does not require an existing network, ii) enables instant discovery and full coverage of the wireless environment, iii) avoids the overhead of continuous information pushing, iv) mitigates the constraints of wireless protocols in pushing relevant information, and v) accounts for the simultaneous execution and the sheer number of mobile applications that require discovery. Similar to content centric networking *in existing* wired [16] and wireless [17] networks, SO-Fi enables users and applications to build communication on content rather than physical locations that hold data. However, these approaches assume connectivity to a transport substrate, i.e., a wireless network or the Internet, to be available. In contrast, device-to-device networking in pervasive mobile scenarios needs to, in the first place, establish a communication substrate between devices, e.g., an 802.11 network. Addressing this issue and solving the intuitive problem of how to identify and connect to the desired devices in mobile scenarios, SO-Fi then makes two contributions:

Content-centric wireless networking: SO-Fi broadcasts content discovery queries in the SSID of 802.11 Probe Request frames of the 802.11 association process. We modify the 802.11

access point (AP) functionality to opportunistically originate 802.11 networks as a results at devices that provide content matching the request. Corresponding provider responses in Probe Response frames then trigger the association of the requesting client to this network, providing a communication infrastructure. In this, hash function-based content addressing adheres to the space constraints of 802.11 frames.

Security in content discovery and provision: SO-Fi incorporates use-case-specific security credentials to provide discovery confidentiality in the observable wireless medium. We support authentication for access control and derive WPA2 keys per-request to secure network access and encrypt traffic. To protect providers from Denial-of-Service (DoS) attacks, SO-Fi employs cryptographic client puzzles.

As an illustrating example, imagine the two motivating scenarios in *InSight* [18], namely visually identifying i) persons with similar interests, e.g., that want to share a cab at the airport, by way of an annotated message and ii) personal friends against a group of people by way of visual fingerprints. Traditionally, in the absence of a global coordination entity as in [14], [15], each person would need to establish an 802.11 network or associate to an existing one in order to transmit fingerprints. The lack of coordination and lack of content semantics in network establishment and associations results in a random number of 802.11 networks, confining users to each network and inducing overhead when iterating through all networks. In contrast, the *InSight* application would, using SO-Fi, encode a query for *InSight* users. The resulting on-demand 802.11 network then covers only the relevant devices and facilitates the exchange of visual fingerprints and messages between application instances. Conversely, when seeking and identifying personal friends, along with sharing and displaying sensible information, *InSight* would encrypt the SO-Fi query with a shared secret. This authenticates the request towards the friend, triggering fingerprint and information transmission in a secure 802.11 network between requester and provider.

We detail content-centric wireless networking in Section II and the incorporation of security mechanisms in Section III. Section IV evaluates SO-Fi in terms of feasibility, overhead, and real-world applicability. Section V discusses related work. We conclude in Section VI.

II. Content-centric Wireless Networking

In current approaches, interactions among devices are constrained by the network infrastructure boundaries, as clients become aware of the accessible content only after associating to a specific network. Instead, we strive to enable a communication paradigm where network associations are a consequence of the content provided by individual devices. In SO-Fi, we thus focus on enabling pervasive and instant content discovery, where clients can query all devices in communication range, independent of already established network infrastructures. Upon a positive response to a query, the same discovery procedure triggers the 802.11 association process to establish an on-demand communication channel between requester and provider. To uniformly address content in the discovery and provision process, SO-Fi requires a consistent addressing scheme. Content-centric networking then provides several benefits over the network-centric paradigm in 802.11.

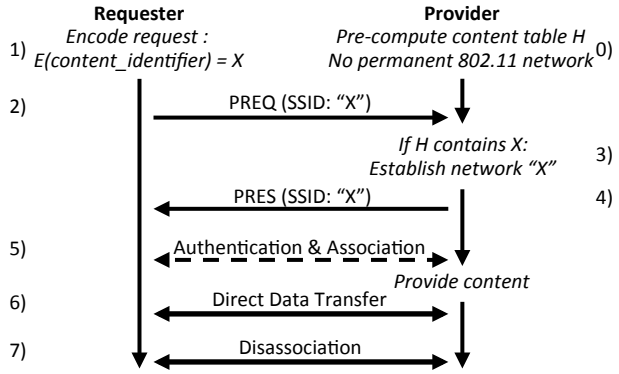

Fig. 2. Content-centric on-demand wireless discovery and provision. Requests are encoded in SSID fields of Probe Request frames using an encoding function `E()`. Providers offline pre-compute an index table H of the content they provide, compare observed requests against this table, and establish a network dedicatedly for this request.

A. Content-centric On-demand 802.11 Wi-Fi

SO-Fi targets instant and full coverage of the pervasive discovery scope, while avoiding the overhead of iteratively associating to the possible multitude of accessible networks. At the same time, SO-Fi aims at efficiently establishing a network infrastructure and subsequent association as soon as a provider for the requested content is identified. To this end, we embed content discovery queries and replies in the request-response sequence of the ubiquitously supported 802.11 association process, ensuring real-world applicability.

The original 802.11 association process requires clients to query for available networks by sending Probe Request frames (PREQ) with either a specific network identifier (SSID) or an empty (wildcard) one. Devices (APs) already providing a network with a matching SSID respond with a Probe Response frame (PRES), carrying the necessary information for the client to authenticate and associate to the network.

SO-Fi moves away from predefined network roles for devices as either clients or APs, enabling devices to dynamically establish network infrastructures around content discovery. Specifically, discovery is performed by encoding content requests into the SSID field of PREQs; after matching the indicated SSID against the locally available content, content providers announce and make accessible a dedicated 802.11 network. We embed this process into the original 802.11 mechanism, cf. Figure 2 (italic font), along three phases: content request, network establishment, and content provision.

To this end, a user-space application encodes the content request into the SSID field, e.g., `E(ubicomp2012) = X` (Figure 2, step 1). We detail the design of the encoding function `E()` in the next section. The application then starts the native 802.11 association process, i.e., triggers a PREQ for a network with the encoded SSID `"X"` (step 2). Using the same encoding function `E()` as in step 1, devices (offline or continuously) pre-compute a content table H that maps (multiple) encoded keys to the locally provided content items (step 0), i.e., data, users, or applications. Upon reception of a query, this enables providers to efficiently look up the encoded request `X` in H.

To respond to a request, SO-Fi extends 802.11 AP functionality with the ability to establish a network with a specific SSID, e.g., `"X"`, on-demand upon reception of a PREQ and

a positive check for X in H (step 3). Resuming the standard 802.11 association process, establishing a network "X" entails transmission of a PRES with SSID "X" (step 4). Reception of this PRES then triggers the 802.11 authentication and association process at the client (step 5)[1].

In this network, actual content transmission (step 6) occurs as traditionally done with regard to the content type, e.g., a direct item download [7] or exchanging social network profiles [6]. As both the provider and requester are aware of the content type, the requesting device can proactively boot the proper application, e.g., a browser, while the provider prepares the demanded service.

B. Content Encoding

The query-response mechanism in SO-Fi depends on a mechanism to address content consistently across distinct devices, analogous to using unique resource identifiers (URIs) [19] or feed and content IDs [20]. However, these approaches build on an existing network infrastructure and higher layer solution, e.g., HTTP and DTN Bundle protocols, that allows complex container structures, e.g., XML or ASF, to identify content. Instead, SO-Fi embeds content requests in the SSID field of Probe Request frames, entailing as a consequence a space constraint of 32 bytes and printable characters in Ascii85 encoding.

While content identifiers may take the form of very diverse and complex formats, SO-Fi currently uses file and service names as well as layer 2 addresses and public keys to identify devices and users, respectively. To meet the aforementioned limitations, SO-Fi uses these identifiers as input for a cryptographic hash function, e.g., SHA-1, to derive unique content identifiers of limited length. The application subsequently transforms the hash value via an Ascii85 encoder.

To perform the actual encoding, hash functions lend themselves to the needs of SO-Fi as they naturally adhere to space constraints with a typical output length of 20 bytes for arbitrary input values. We use the remaining space in the SSID field to transport additional flags, e.g., a unique request ID and security combinations, in requests and responses.

C. Applicability and Benefits

SO-Fi enables users and applications to "demand" the creation of an 802.11 network at devices in transmission range without incurring time or communication overhead, i.e., in a one-step network and content request. Requests can be executed when triggered by the user or periodically in the background to autonomously interact with fellow application instances. Comprehensively covering the pervasive wireless discovery scope, content-centric networking then naturally and immediately pairs devices, i.e., users and their application instances, in a conventional 802.11 network.

SO-Fi networks then support communication and exchange of arbitrary content as envisioned by the application. For example, networks resulting from plain file lookups, such as documents or multimedia, may provide a transport substrate on layers 2 and 3, while the actual content transfer occurs over HTTP. We demonstrated this variant in a preliminary design at MobiCom'12 [21]. Alternatively, applications may plug in their own communication mechanisms on top of 802.11, e.g., link-layer content discovery and TCP-based publish/subscribe content exchange [20] or event-driven content synchronization [6] once devices are in a common network.

[1]In case of responses by multiple providers, 802.11 implementations choose the response with the highest signal strength. Custom implementations allow further information, e.g., device load, inside Information Elements in PRESs.

TABLE I. SECURITY FEATURES IN SO-FI DERIVED FROM SUCCESSIVELY HASHING THE CONTENT IDENTIFIER IN COMBINATION WITH GROUP OR USER KEY k.

Hashing step	Use	Security Feature
E^1 = E(content_identifier) $\oplus$ k	WPA2 PSK	Network access control, traffic encryption
$E^2 = E(E^1) \oplus k$	Cryptographic client puzzle	Provider-side DoS protection
$E^3 = E(E^2) \oplus k$	Content request in SSID field	Obfuscation/Encryption of initial request

III. SECURITY IN CONTENT DISCOVERY AND PROVISION

SO-Fi widens the scope of pervasive content discovery to all devices in transmission range. Therefore, it already provides a self-contained mechanism for comprehensive discovery and provision of *public* content. Applications can build on SO-Fi to, e.g., offer tourism information at public places or navigation and information services at airports, malls, etc.

However, this basic mechanism does not yet meet the requirements of discovering and providing *sensitive* content that is restricted to specific users or groups within an application, e.g., in mobile social networking [6] and peer-to-peer [20] or localized content sharing [7]. While the participating parties are able to achieve unlinkability [22] by randomizing their Layer 2 addresses for each new request and response, respectively, we identify four security aspects of SO-Fi, that need to be addressed. Namely, secure content discovery and provision requires i) authentication, to allow for differentiated treatment of sensitive content, ii) confidentiality of requests and responses, iii) 802.11 network security to control network access and secure content transmission, and iv) DoS robustness, to protect providers from replayed requests that exhaust their resources.

A. Credentials and Authentication

Sensitive content in the form of confidential data or restricted applications requires differentiated treatment in discovery and provision. SO-Fi thus supports inclusion of user and group keys, that are pre-established offline, in the request and provision mechanism, as unavoidable in secure password, public key, and certificate-based communication [22]–[24]. We discuss the absence of secure keys, i.e., in a fully spontaneous scenario as in [12], [13] in Section III-D.

Secure keys can be application-specific keys, passwords, or iterative one-time authentication tokens, e.g., cryptographic hash-chain elements. SO-Fi thereby leverages the semantic relation between users of sensitive content to bind keys to specific content, as manifested in, e.g., the installation of a common mobile application or prior trust relations in social networks. Orthogonal to our current design, recent advances in wireless key establishment [25] might allow extending our solution to fully spontaneous communication.

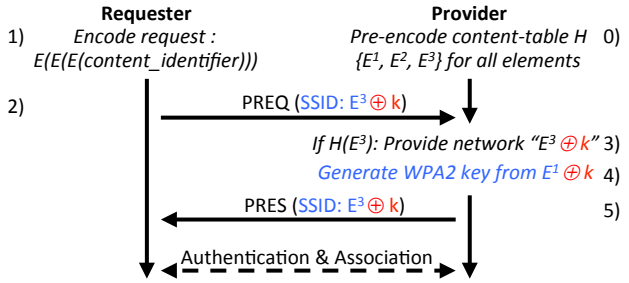

Fig. 3. Secure network provision and content confidentiality in SO-Fi. E^1 is not computable from overhearing $E^3 \oplus k$ due to the pre-image resistance of the applied cryptographic hash function.

Providers thus associate (multiple) *(mac_address, key)* tuples[2] with content. Authentication then seamlessly integrates with the previously described mechanism. A requester can XOR the content request X in PREQs with the shared key (Figure 2, step 2) to authenticate the request. The provider matches the Layer 2 address in the PREQ to a stored tuple and, if positive, reconstructs X as (SSID $\oplus$ *key*). If the tuple *(mac_address, key, X)* then matches a stored tuple, the provider responds to the request and otherwise ignores it.

In the following, we support inclusion of given user or group keys in requests and exploit the pre-image resistance of the hash function E() used in content encoding to enable confidentiality, 802.11 network security, and DoS robustness. Providers and requesters can generate keying material for these functionalities by hashing content identifiers to generate the sequence $E^1 = E(content\_identifier)$, E^2, and E^3 in combination with a key k. Encrypting each hash value with the key k thereby allows transmitting requests and responses in the open wireless medium; attackers are then unable to derive previous hash values and associated content identifiers. We use distinct hash values E^1, E^2, and E^3 to hide k, to separate the respective functionality in a clear design, and allow devices to differentiate the requested functionality.

Table I shows the specific features enabled by single steps in the hashing sequence. In Figures 3 and 4, blue text highlights the respective security functionality, while use of the key k is marked in red. Please note that all functionalities are combinable to accommodate the requirements of requesters, providers, and applications. Providers can signal the specific combination inside PRES frames.

B. Content Confidentiality and 802.11 Network Security

Even after securing requests using user keys, content provision in on-demand 802.11 networks entails two distinct security risks. First, an eavesdropping attacker observing requests and corresponding 802.11 networks can replay successful requests, i.e., the PREQ and used MAC address, to obtain the requested content, breaching content *confidentiality*. Second, an actively scanning attacker can discover on-demand created 802.11 networks. SO-Fi thus demands *802.11 network security* to secure network access and encrypt actual content transmissions.

To fulfill both security requirements, devices in SO-Fi use the hash value $E^1 \oplus k$ to generate a custom WPA2 PSK and achieve state-of-the-art 802.11 network security. To generate a WPA2 key, $E^1 \oplus k$ serves as the input to the 802.11 PBKDF2 function, instead of a given network passphrase and the network SSID as in unmodified 802.11. In addition, $E^3 \oplus k$ encrypts the transmitted content identifier. Figure 3 shows the embedding of both into the query-response sequence of SO-Fi.

[2]Pre-defined, random, or cryptographic Layer 2 address, using the respective credentials of requester and provider [22] on a per-application basis.

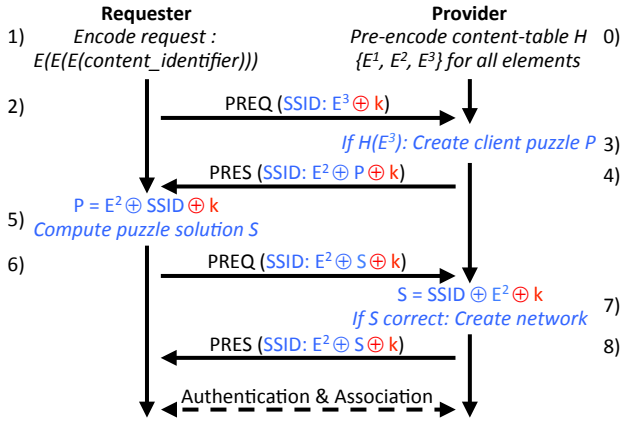

Fig. 4. Provider-side DoS protection using cryptographic client puzzles.

The pre-image resistance of the hash function, in combination with the used key, prevents devices from computing the original content identifier, and thereby the requested content, from the overheard transmissions. Similarly, it is not possible to derive the 802.11 WPA2 network key generated using $E^1 \oplus k$ from $E^3 \oplus k$.

Content requests and provision are thus *confidential* because content identifiers cannot be derived. Furthermore, an attacker using $E^3 \oplus k$ as the content request in a replay attack lacks the 802.11 network key generated from $E^1 \oplus k$. SO-Fi thus supports state-of-the-art 802.11 network security through per-demand WPA2 network keys and the subsequent traffic encryption using client-specific session keys.

C. Provider-side DoS Protection

In SO-Fi, each successful request causes providers to expend resources by creating a network in expectance of a benign client. Even without actually accessing the network, malicious devices can thus mount a *Denial-of-Service* (DoS) attack by replaying successful requests to deplete the resources of providers.

To enable DoS protection for providers, SO-Fi incorporates the proven method of *cryptographic client puzzles* [26]. In this approach, a providing device, e.g., a web server, requires the requesting device to prove the benign character of its request by expending resources itself, prior to requesting resources from the provider. Analogous, a provider in SO-Fi can require a requester to solve a cryptographic puzzle P of adjustable difficulty. E.g., puzzle difficulty can be adjusted to the frequency of a specific request. The puzzle solution S then is required as input to the request for which the provider actually establishes a network. We refer to the original paper [26] for details on creating and solving client puzzles as well as the security properties of this approach.

Figure 4 shows the embedding of client puzzles in SO-Fi. Instead of establishing a network immediately when receiving $E^3 \oplus k$, the provider generates a cryptographic puzzle P (step

3) according to [26] and sends a PRES with SSID ($E^2 \oplus P \oplus k$), the communication ID of the PREQ and a flag indicating the puzzle difficulty (step 4). The ID allows the requester to identify this response, extract *P* as ($E^2 \oplus$ SSID $\oplus k$), and compute its solution *S* (step 5). A PREQ with SSID ($E^2 \oplus S \oplus k$) (step 6) proves the correct solution to the provider and triggers the actual creation of a network for content provision (step 7, 8).

Devices overhearing clear-text puzzles could compute the solution faster than the original requester and thereby highjack the association process in a Man-in-the-Middle attack. To prevent this, we encrypt the puzzle *P* and the solution *S* using E^2 and k in steps 4, 6, and 8. For this, we treat the random values *P* and *S* as one-time pads in combination with E^2. We use the XOR function to allow for simple retrieval of input values and faster computation in comparison to encryption functions. SO-Fi never transmits *P*, *S*, E^2, or *k* in clear text, preventing attacks by uninvolved devices.

Cryptographic client puzzles alleviate repeated malicious content requests and overloading of provider devices in SO-Fi. However, a PRES injected in step 4 with an overheard communication ID may prompt the requester to solve a random puzzle contained in the SSID. To prevent such DoS attacks on requesters, 2 bytes of each transmitted puzzle carry a pre-defined checksum covering the rest of the puzzle. As E^2 is not known to uninvolved devices, pre-computing or guessing puzzles that produce a correct checksum after ($E^2 \oplus$ *SSID* $\oplus k$) is infeasible, protecting requesters from DoS attacks.

D. Security Considerations

We argue that, under the assumption that the occurrence of content identifiers in searches is distributed equally over the content name space, the proposed mechanisms establish confidential and secure content discovery, provision and access, even without pre-established keys. This is because the pre-image resistance of the hash function prevents overhearing devices from computing E^1 or E^2 from E^3. Exclusively pre-computing all possible *(content_identifier, (E^1, E^2, E^3))* tuples would enable overhearing devices to match an observed request E^3 to the original content identifier. Specifically, it is not possible to derive from E^3 the 802.11 WPA2 key generated using E^1 or the cryptographic puzzle *P* encrypted with E^2. We thus regard the inclusion of k as optional with regard to the application requirements.

In practice, however, it is possible to precompute offline the tuples (E^1, E^2, E^3) for popular search terms, e.g., "Internet Access" or "Tourism Information". Malicious devices would thereby be able to identify requests and subsequently eavesdrop on the network traffic. Applications without pre-established keys thus need to employ higher-layer encryption, e.g., via TLS, to secure data transmissions in the absence of network security. Specifically, applications may be discoverable via unsecured SO-Fi interactions but employ strong encryption and authentication of messages to protect message and application confidentiality. SO-Fi thereby supports secure pervasive content discovery and provision that is adjustable to application and provider demands and combinable with higher-layer approaches.

IV. Evaluation

In this section, we evaluate the content discovery feasibility, performance, and applicability of SO-Fi, as well as the overhead induced by the proposed security mechanisms. To this end, we show that content discovery using on-demand 802.11 network associations causes negligible time overhead, while removing the need for network selection and subsequent discovery protocols. We quantify the overhead of creating content tables *H* of different size and show that establishing key and puzzle material via subsequent hash operations only adds minimal overhead. We evaluate the performance of current mobile devices in computing the required 802.11 WPA2 PSK and show the scalability and performance of the proposed cryptographic client puzzle mechanism. Last, we show the adjustable time complexity of the proposed security mechanisms in SO-Fi.

We implemented the request functionality of SO-Fi on Linux- and Android-based mobile devices. We realized the on-demand provider functionality in the popular *hostapd* software package, thereby enabling deployments on Linux-based mobile devices and Wi-Fi APs as well as Android devices. As Linux devices, we use Lenovo S10-3 Ideapads with dual-core 1.5 GHz CPUs and Atheros AR9285 802.11n wireless cards, representing commodity mobile netbook devices. In our evaluation, the netbooks serve as SO-Fi providers. As Android devices, we use both the Samsung Nexus S and Samsung Galaxy Nexus devices to show the impact of both CPU processing power in newer devices and different wireless chipsets. While the Nexus S builds on a 1 GHz CPU and Broadcom BCM4329 802.11n wireless chipset, the newer Galaxy Nexus includes a dual-core 1.2 GHz CPU and BCM4330 802.11n wireless chipset. In computation-based evaluations, we also provide results for a 2.93 GHz quad-core Intel i7 Linux PC for comparison.

A. Content Discovery Performance

Embedding content discovery and on-demand network provision into the 802.11 association process extends the 802.11 AP functionality of the provider. Namely, instead of immediately responding to relevant Probe Requests with a Probe Response, providers in SO-Fi first perform a lookup in the content table for the given request and subsequently instantiate the corresponding 802.11 network. We thus evaluate the feasibility of complete content discovery within the message and time constraints of the 802.11 association process by evaluating the time requirements of on-demand 802.11 network associations. In SO-Fi, this represents the entire time requirement of content discovery, network provision, *and* association to a common network, whereas approaches using traditional 802.11 require this association time *and* subsequent discovery time overhead for each potential network.

We evaluate the association time of both traditional, i.e., permanent, and on-demand 802.11 networks in SO-Fi. On average, SO-Fi devices respond to a request, i.e., establish an 802.11 network and send a PRES, within 67 ms. Figure 5 shows the average time and standard deviation of 100 associations for different client (requester) and AP (provider) devices. Regarding the evaluated AP devices for each client device, we deduce the feasibility of our design as SO-Fi only introduces a marginal overhead of up to 0.2 s, if any at all. The results further highlight the impact of different Wi-Fi chipset-driver

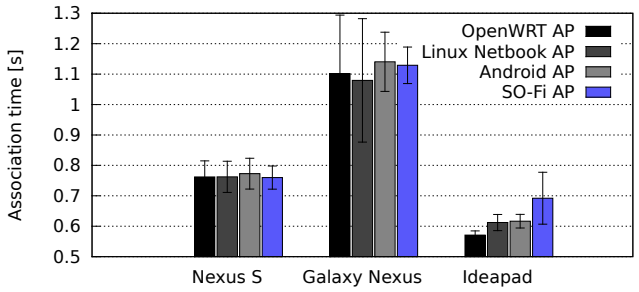

Fig. 5. Average association time to unmodified 802.11 networks and SO-Fi networks using different client and AP devices. In comparable time, SO-Fi provides content discovery, content lookup, and on-demand network provision and avoids network selection and subsequent content discovery overhead.

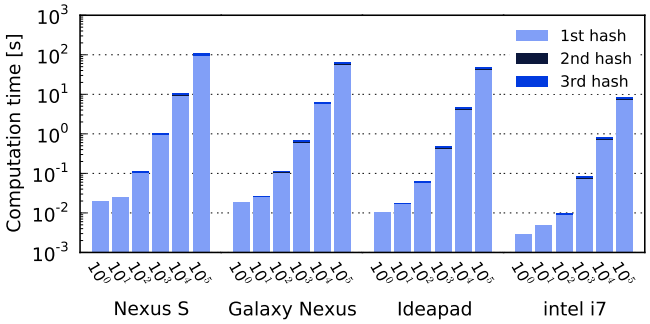

Fig. 6. Average computation time of multiple hash operations in content tables of increasing size. Hash values serve as input for WPA2 network keys, client puzzles and content identifiers. Please note the logarithmic scale.

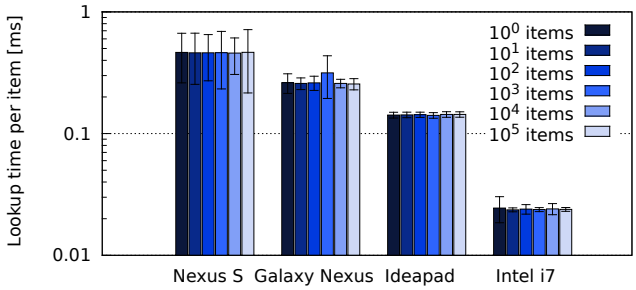

Fig. 7. Average lookup time for one content item on different devices with regard to the number of stored items. Lookup times are below 1 ms and are constant per devices, supporting the feasibility of per-request lookups. Please note the logarithmic scale.

combinations on devices that run the same operating system, e.g., Nexus S and Galaxy Nexus, but use a Wi-Fi driver and medium access functionality built mainly in chipset-specific firmware and Linux-generic software, respectively. Please note that the success rate of associations in SO-Fi, 100 % in our experiments, is equally affected, e.g., by mobility and obstacles, as traditional 802.11. Also, content discovery in SO-Fi appears as a standard-compliant network association process to legacy 802.11 networks and hence does not induce any conflicts or performance degradation.

As the main result of these measurements, SO-Fi enables discovery[3] of arbitrary content, i.e., data, applications, and persons, within the communication and time overhead of *a single 802.11 association*. With regard to Pitkänen [5], SO-Fi combines service *and* device discovery in one step as providers answer both the request for content and for a providing device in the PRES. SO-Fi thereby avoids the overhead of network associations that fail to yield the requested content, as well as dedicated service discovery protocols subsequent to the association. This minimal time and communication complexity especially benefits users in mobile scenarios with time-constrained device encounters. SO-Fi thereby enables ubiquitous user-triggered or background querying of all devices in transmission range at any point in time, independent of network infrastructures or multi-hop communication (immediate or time-decoupled).

B. Content Encoding and Lookup

In SO-Fi, requesters compute the hash sequence (E^1, E^2, E^3) to perform a request and providers need to establish the content table H. Furthermore, providers have to perform a lookup in the content table to find out whether it serves an observed request. In this section, we evaluate our design of using consecutive hash operations to generate requests and the content table and show the practicality and scalability of the proposed provider functionality.

To illustrate the computation time at requester and provider, Figure 6 shows the average time for 1000 calculations of 1, 2, and 3 hash values for increasing numbers of content items on Android devices, a Linux netbook, and an Intel i7 computer for comparison. Please note, the time difference between the first, second and third iteration is indiscernible for the 10^0 and 10^1 data points. The results support our design in four aspects.

First, computing a single hash sequence of 3 hash values (Figure 6, 10^0 data points), as required to generate all values for a request, induces negligible time overhead on all devices. Generation of a request is thus possible instantaneously and with low computational overhead.

Second, Figure 6 shows the feasibility of our design from a provider perspective. While the time required to encode the content table H rises with the number of items, regardless of the provider device, this computation can be performed in the background. Already computed entries can then be provided while further computation is performed. Building the content table in SO-Fi thus keeps the device available for other tasks and allows for a seamless integration of additional content items. Furthermore, for realistic numbers of content items (e.g., $\leq 10^4$) computing the content table takes less than 10 s on all devices, underlining the feasibility on real-world devices.

Third, we evaluate the overhead of additional hash operations to establish E^1, E^2, and E^3 for security functionality. As Figure 6 shows, computing additional hash values for any number of content items induces marginal overhead in comparison with the initial hashing and I/O overhead that is required to create the original content table entry. The generation of keying material for security functionality thus seamlessly integrates into the computation of both single requests and content tables at negligible cost.

[3]Note that the measurements only represent content discovery. *Content access* and *transmission* depend on factors on top of SO-Fi, such as the transport protocol and actual application. Especially, the results do not include DHCP durations.

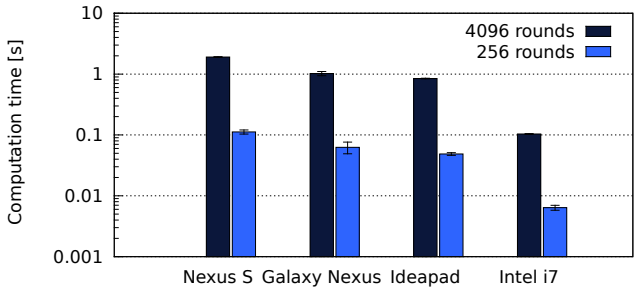

Fig. 8. Computation time for WPA2 PSK (802.11 PBKDF2). The high entropy of passphrase E^1 and SSID E^2 or E^3 allows devices to reduce the number of iterations from 4096 to 256. Please note the logarithmic scale.

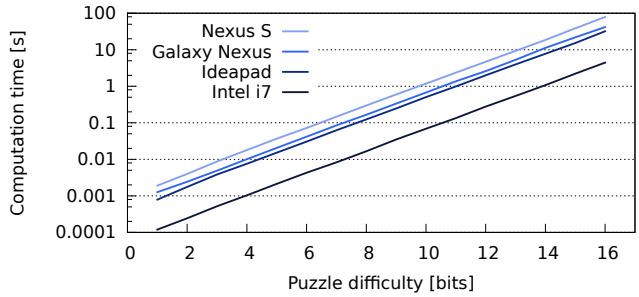

Fig. 9. Exponential time complexity of cryptographic client puzzles with regard to the puzzle difficulty. Scaling the puzzle difficulty allows adjusting the required client resources. Please note the logarithmic scale.

Last, Figure 7 quantifies the time it takes to look up a single item, as necessary when observing a request, in content tables of increasing size on different devices. As expected, standard implementations of, e.g., hash tables, require constant lookup times on all devices, regardless of the number of stored items. Using available content table implementations, providing devices can thus respond to requests within the time constraints of the 802.11 association process.

C. WPA2 PSK Computation

In contrast to permanent pre-configured wireless networks, devices in SO-Fi compute the WPA2 PSK for a network on-demand. Securing SO-Fi networks thus induces a time overhead in the discovery and association process and influences the applicability of SO-Fi with regard to time-constrained communication opportunities and user-perceived usability.

The 802.11 standard [27] mandates the use of the PBKDF2 procedure to derive pairwise shared keys from a given network passphrase and the network SSID. To balance the relatively low entropy of typical, human-readable passphrases and SSIDs, 802.11 AP implementations compute keys over 4096 PBKDF2 iterations. In order to quantify the overhead of this computation, we implemented this procedure in a stand-alone Python script and performed 1000 distinct calculations. As shown in Figure 8, computing 4096 iterations requires around 1 s on typical mobile devices, i.e., almost the time required for the actual SO-Fi association.

The number of PBKDF2 iterations thereby induces a tradeoff between the key security and the computation time. In SO-Fi, the hash-based, high-entropy identifiers E^1 and E^2 or E^3 serve as hidden, random input to PBKDF2, increasing input entropy compared to human-readable passphrases. Given the short lifespan of networks in SO-Fi, devices can thus choose to trade shorter computation time for a derived key that is less secure. Figure 8 quantifies the gain in computation time when reducing the number of iterations to 256. Requiring 0.1 s or less on current devices, this configuration significantly reduces the impact on the discovery time. Devices in SO-Fi thus can indicate the required number of iterations in content request and response flags to choose the appropriate degree of security.

D. Scalable DoS Protection

To protect providing devices from DoS attacks, we propose the use of cryptographic client puzzles. By adjusting the puzzle difficulty, i.e., the required bit length of the solution, providers can adjust the required computational effort to their current state and the requested content. A provider that is under attack or observes high request frequencies for a specific content item can pose high difficulty puzzles, while low to medium puzzle difficulties are otherwise sufficient.

We thus evaluate the complexity of solving cryptographic puzzles with increasing difficulty on typical current mobile devices to give an indication of appropriate puzzle difficulties. To allow a comparison with the previously evaluated content discovery and network association times in SO-Fi, Figure 9 shows the average time overhead of solving 1000 puzzles of each difficulty. The results indicate that, for current mobile devices, a puzzle difficulty of 10 bits with an average computation time of 1 s can serve as a lower bound. On the other hand, difficulties higher than 14 bits induce significant, probably unacceptable delays in the discovery.

The increasing computational power of mobile devices, hinted at by the Intel i7 CPU results, may require increasing puzzle difficulties in the future. SO-Fi currently supports a maximum puzzle difficulty of 32 bits. The exponential growth of computation time thereby affords adjusting puzzle difficulties to increasing CPU capabilities of mobile devices.

E. Secure Content Discovery Performance

To secure content discovery in SO-Fi against replay attacks, eavesdropping and denial-of-service attacks, we proposed generating WPA2 PSK key material and cryptographic client puzzles from content requests and user keys. As shown, the overhead of generating the key material itself during request or content table encoding is negligible. However, calculating the actual WPA2 PSK and the solution to a client puzzle requires additional computation during the discovery process. Similar, the challenge-response mechanism of client puzzles requires an additional protocol step to exchange the puzzle and the solution, respectively. In this section, we thus evaluate the detailed timings of secure content discovery in SO-Fi. Please note that we do not dedicatedly evaluate user authentication because the protocol steps are identical to the evaluated mechanisms and the computational overhead of the `XOR` function is negligible.

Figure 10 shows the cumulative average timing of 100 SO-Fi associations for different requesting devices with regard to the activated security mechanisms. For a better illustration of the overhead, we measure the duration of the respective protocol steps, as observed at the requesting device. From our cryptographic puzzle evaluation, we set the difficulty to 10 bits

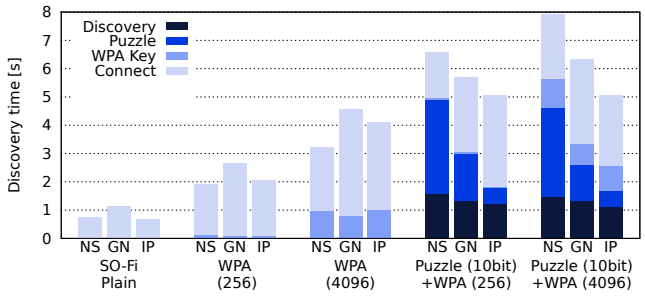

Fig. 10. Client-side cumulative timings for secure content discovery in SO-Fi with regard to the active security mechanisms for Nexus S (NS), Galaxy Nexus (GN) and Lenovo Ideapad (IP) Android and Linux devices, respectively.

and evaluate the calculation of WPA2 PSKs over both 256 and 4096 PBKDF2 iterations. Please note that the evaluated time overhead for cryptographic puzzles on Android devices varies from the previously presented stand-alone results. This is due to performance differences in the stand-alone Python implementation and the Java-based Android implementation used in our real-life prototype.

The time for protocol steps that depend on device-specific Wi-Fi performance and scanning behavior, i.e., the initial "Discovery" step (Figure 4, steps 2-4) and the "Connect" step[4] (Figure 2-4, authentication and association), slightly varies between the requesting devices. In our implementation, we adhere to the standard Android and Linux API interfaces and tools, especially *wpa_supplicant*, to provide an indication of the practical performance of SO-Fi. This performance can be improved, especially by optimizing the wireless scanning behavior of the respective OS and driver.

With regard to the computational capabilities of the evaluated requesting devices, Figure 10 shows that the proposed security mechanisms allow adjusting the security overhead depending on the requested content and the state of the providing device. Although we only provide measurement results for 10 bit puzzle difficulty as well as 256 and 4096 PBKDF2, SO-Fi supports all intermediate settings. However, 4096 PBKDF2 iterations, i.e., the number of iterations proposed in the 802.11 standard, only require 1 s on current devices. As upcoming devices will decrease this computation time, the difficulty of cryptographic puzzles lends itself to be the main scaling parameter in SO-Fi.

Figure 10 furthermore shows the discovery time for different provider-side settings and applications. In attack-free situations without the need for DoS protection and with 256 PBKDF2 iterations, secure content discovery takes less than 2.5 s. Increasing the number of PBKDF2 iterations, as mandated in the 802.11 standard, results in timings of less than 4.5 s. An application for these settings is public, resource-friendly or insensitive content, e.g., in touristic settings, that is freely available. Last, content discovery only requires up to 8 s in case of providing devices under attack or sensitive requested content. By scaling the puzzle difficulty, this time requirement can be adjusted to the situation and the requested content. Examples for such protected content discovery and provision are resource-heavy content and content provision by resource-constrained devices. Last, augmenting discovery by authentication allows further restricting content access to trusted devices and users.

As the main result, the measured times support practical content discovery of SO-Fi within encounter durations in realistic mobile scenarios as evaluated, for example, by Pitkänen [5]. To model a challenged network scenario, we regard a radio range of (only) 30 m and a discovery time of 8 s in SO-Fi, including 4096 PBKDF2 iterations and a 10 bit puzzle difficulty. Given these parameters, the vast majority of contacts in both the trace-driven KAIST and the synthetic HCS evaluation allows for content discovery *and* subsequent content access, regardless of the scan interval. When assuming a discovery process without DoS protection and with only 256 PBKDF2 iterations, i.e., a discovery time of less than 2.5 s, virtually all contacts are usable. SO-Fi thus realizes content discovery with viable time and computation requirements in urban scenarios.

V. Related Work

SO-Fi relates to approaches in content-centric networking, wireless protocol overloading, and mobile content discovery.

A. Content-centric Networking

Content-centric Networking (CCN) [16] approaches propose a clean-slate Internet approach that builds addressing, lookup, and routing around content instead of hosts. Similar, wireless content-centric approaches [17] do not target the establishment of network structures based on content but assume existing network structures in which user traffic centers on retrieving content. In mobile scenarios, however, the first step to communication in the network-centric design of 802.11 is the discovery of and association to an existing network. SO-Fi thus differs from the above mentioned approaches in catering to and building this first step around the availability of content of interest, instead of resolving and routing traffic.

B. Wireless Protocol Overloading

Beacon stuffing [10] overloads 802.11 Beacon frames to support network selection and wireless advertisements. Similarly, APs in the 802.11*u* standard [28] broadcast additional network information to support network selection. Davies et al. [11], *E-Shadow* [12] and *E-Smalltalker* [13] overload Bluetooth device names, the Bluetooth Discovery Protocol, and 802.11 Beacon frames, respectively, to push application information, semantically enriched device names, and interests.

In contrast to SO-Fi, the presented approaches are *push-based*, i.e., they proactively and continuously signal information. The limited space in wireless frames thereby prevents pushing all available content. Conversely, SO-Fi realizes request-driven, space-efficient content discovery and access in a dedicated network. The only request-driven approach that we are aware of is the Access Network Query Protocol (ANQP) in 802.11*u* [28]. However, clients in ANQP can only query for characteristics of *pre-defined available* networks.

C. Pervasive Mobile Content Discovery

Haggle [29] provides peer-to-peer exchanges based on interests, similar to publish-subscribe designs [20]. Both require an existing network for requester and provider, as do network-wide

[4]Includes the time for provider-side WPA2 PSK calculation.

content discovery [30] and push-based service notifications [31]. In contrast, SO-Fi is the first approach to enable content discovery outside the narrowness of established networks. In this, SO-Fi greatly reduces communication overhead and adheres to viable discovery times while simultaneously widening coverage to *all* nearby devices. Existing *secure* approaches [23], [24] also require pre-established trusted network infrastructures as well as a trusted (global) directory. SO-Fi supports use-case-driven confidentiality and security without any trusted infrastructure.

DataSpotting [15] and *ICON* [14] discover and pair devices by way of Internet-scale match-making between requests and provided content. In contrast, SO-Fi requires neither an Internet uplink nor a global database but emphasizes direct device-to-device communication.

VI. Conclusion

We proposed SO-Fi, ubiquitous content-centric wireless networking with use-case-specific security that provides intuitive discovery and connection establishment in mobile networks. Removing the constraints of pre-established network infrastructures enables a comprehensive discovery scope for mobile applications. In addition, network infrastructures are established to provide content only on-demand, making content-centric networking efficient. Completing our preliminary design [21], the approach seamlessly integrates adjustable security mechanisms to achieve practical 802.11 network security, content and traffic confidentiality, and DoS robustness. Our evaluation of SO-Fi shows its feasibility and limited overhead. We make the source code accessible [32] for developers.

As a result, SO-Fi provides a novel building block for mobile applications and offloading approaches. Applications thereby can exploit SO-Fi to specify user- or event-triggered or continuous background discovery. Building on locally comprehensive discovery scopes, visualization techniques may incorporate content availability with user contexts, e.g., the current position. The integration of similarity preserving hashes [33] as encoding functions promises fine-tuned requests based on domain knowledge. Attribute-based encryption [34] provides a fine-tuned access control mechanism based on a priori established (group) attribute. We aim to investigate the integration of these mechanisms in future work.

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) and the DFG Cluster of Excellence on Ultra High-Speed Mobile Information and Communication (UMIC).

References

[1] A. Petz, A. Lindgren, P. Hui, and C. Julien, "Madserver: a server architecture for mobile advanced delivery," in *CHANTS*, 2012.

[2] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, "Cellular traffic offloading through opportunistic communications: a case study," in *CHANTS*, 2010.

[3] Y. Li, G. Su, P. Hui, D. Jin, L. Su, and L. Zeng, "Multiple mobile data offloading through delay tolerant networks," in *CHANTS*, 2011.

[4] S. Liu and A. D. Striegel, "Exploring the potential in practice for opportunistic networks amongst smart mobile devices," in *MobiCom*, 2013.

[5] M. Pitkanen, T. Karkkainen, and J. Ott, "Mobility and service discovery in opportunistic networks," in *PerCom Workshops*, 2012.

[6] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot, "Mobiclique: middleware for mobile social networking," in *WOSN*, 2009.

[7] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju, "Floating content: Information sharing in urban areas," in *PerCom*, 2011.

[8] K. Thilakarathna, A. C. Viana, A. Seneviratne, and H. Petander, "Mobile social networking through friend-to-friend opportunistic content dissemination," in *MobiHoc*, 2013.

[9] M. Bakht, M. Trower, and R. H. Kravets, "Searchlight: won't you be my neighbor?," in *MobiCom*, 2012.

[10] R. Chandra, J. Padhye, L. Ravindranath, and A. Wolman, "Beacon-stuffing: Wi-fi without associations," in *HotMobile*, 2007.

[11] N. Davies, A. Friday, P. Newman, S. Rutlidge, and O. Storz, "Using bluetooth device names to support interaction in smart environments," in *MobiSys*, 2009.

[12] J. Teng, B. Zhang, X. Li, X. Bai, and D. Xuan, "E-shadow: Lubricating social interaction using mobile phones," in *ICDCS*, 2011.

[13] Z. Yang, B. Zhang, J. Dai, A. Champion, D. Xuan, and D. Li, "E-smalltalker: A distributed mobile system for social networking in physical proximity," in *ICDCS*, 2010.

[14] H. Wirtz, J. Rüth, T. Zimmermann, and K. Wehrle, "Interest-based cloud-facilitated opportunistic networking," in *CHANTS*, 2013.

[15] X. Bao, Y. Lin, U. Lee, I. Rimac, and R. Choudhury, "Dataspotting: Exploiting naturally clustered mobile devices to offload cellular traffic," in *INFOCOM*, 2013.

[16] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT*, 2009.

[17] G. Alfano, M. Garetto, and E. Leonardi, "Content-centric wireless networks with limited buffers: When mobility hurts," in *INFOCOM*, 2013.

[18] H. Wang, X. Bao, R. R. Choudhury, and S. Nelakuditi, "Insight: recognizing humans without face recognition," in *HotMobile*, 2013.

[19] J. Ott and M. J. Pitkanen, "Dtn-based content storage and retrieval," in *WoWMoM*, 2007.

[20] O. R. Helgason, E. A. Yavuz, S. T. Kouyoumdjieva, L. Pajevic, and G. Karlsson, "A mobile peer-to-peer system for opportunistic content-centric networking," in *MobiHeld*, 2010.

[21] H. Wirtz, D. Martin, B. Grap, and K. Wehrle, "Demo: On-demand content-centric wireless networking," in *MobiCom*, 2012.

[22] 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*, 2008.

[23] F. Zhu, M. Mutka, and L. Ni, "Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services," in *PerCom*, 2003.

[24] F. Zhu, M. Mutka, and L. Ni, "Prudentexposure: a private and user-centric service discovery protocol," in *PerCom*, 2004.

[25] I. Safaka, C. Fragouli, K. Argyraki, and S. Diggavi, "Creating shared secrets out of thin air," in *HotNets*, 2012.

[26] A. Juels and J. G. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks.," in *NDSS*, 1999.

[27] IEEE, "802.11 standard." http://standards.ieee.org/findstds/standard/802.11-2012.html.

[28] IEEE, "802.11u standard." http://standards.ieee.org/findstds/standard/802.11u-2011.html.

[29] J. Su, J. Scott, P. Hui, J. Crowcroft, E. De Lara, C. Diot, A. Goel, M. H. Lim, and E. Upton, "Haggle: seamless networking for mobile applications," in *Ubicomp*, 2007.

[30] M. Pitkanen, T. Karkkainen, J. Greifenberg, and J. Ott, "Searching for content in mobile DTNs," in *PerCom*, 2009.

[31] R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade, "Deapspace: transient ad hoc networking of pervasive devices," *Computer Networks*, vol. 35, Mar. 2001.

[32] "SO-Fi source code." https://github.com/blightzero/SO-Fi.

[33] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala, "Locality-preserving hashing in multidimensional spaces," in *STOC*, 1997.

[34] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy*, 2007.