11-2017

SPICE: Secure Proximity-Based Infrastructure for Close Encounters

Aarathi Prasad
Skidmore College

Xiaohui Liang
University of Massachusetts Boston

David Kotz
Dartmouth College, David.F.Kotz@Dartmouth.EDU

SPICE: Secure Proximity-based Infrastructure for Close Encounters

Aarathi Prasad*
Skidmore College

Xiaohui Liang
UMass Boston

David Kotz
Dartmouth College

ABSTRACT

We present a crowdsourcing system that extends the capabilities of location-based applications and allows users to connect and exchange information with users in spatial and temporal proximity. We define this incident of spatio-temporal proximity as a *close encounter*. Typically, location-based application users store their information on a server, and trust the server to provide access only to authorized users, not misuse the data or disclose their location history. Our system, called SPICE, addresses these privacy issues by leveraging Wi-Fi access points to connect users and encrypt their information before it is exchanged, so only users in close encounters have access to the information. We present the design of the system and describe the challenges in implementing the protocol in a real-world application.

CCS CONCEPTS

• **Networks** → **Network privacy and anonymity**; *Mobile and wireless security*; • **Human-centered computing** → **Smartphones**; *Social content sharing*; • **Information systems** → *Location based services*; *Crowdsourcing*;

KEYWORDS

crowdsensing, smartphones, privacy, anonymity, location, encounters

ACM Reference Format:
Aarathi Prasad, Xiaohui Liang, and David Kotz. 2017. SPICE: Secure Proximity-based Infrastructure for Close Encounters. In *Proceedings of First ACM Workshop on Mobile Crowdsensing Systems and Applications (CrowdSenSys'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3139243.3139245

1 INTRODUCTION

A subset of location-based crowd-sensing applications allow co-located users to contact each other at events [23], or to handle emergency situations [8]. In this paper, we propose a system that allows users to connect not only with co-located users, but also users in spatial and temporal proximity. We define this incident of spatio-temporal proximity as a *close encounter*.

Two devices are in spatio-temporal proximity of each other if they were (i) at the same location almost at the same time, (ii) almost at the same location at the same time, or (iii) almost at the same location at almost the same time. By supporting close encounters, location-based applications can allow users to share and collect information such as images and videos [7] or sensor data [4] from others who were in spatio-temporal proximity. For example, obtaining additional images and videos about a situation from others who were in spatial and temporal proximity can be useful to Alzhemier's patients, event attendees, tourists, forensic investigators, and emergency responders. Spatial and temporal ranges for sharing are application-specific, but we expect "near" (spatial range) to be about 10-50 meters and "close" (temporal range) to be not more than 30 minutes.

Since the users in spatio-temporal proximity may never encounter each other, the naive way for them to share information would be to upload it to a server and allow others to download the information retrospectively from the server. Incorporating a server in the system poses the following challenges: 1) a curious server operator may learn the location histories of the users who upload and download the information, as well as the information itself, and 2) any user, irrespective of their location, may be able to access the information. Prior research, however, has shown that users may be comfortable sharing information only if they see a value in sharing [15], so a user may be concerned if others, who were not in a close encounter with her, for whom the information maybe irrelevant, may be able to access the information, especially if the information was sensitive, such as photos or videos.

To address these challenges, we propose a system called SPICE (Secure Proximity-based Infrastructure for Close Encounters) that leverages existing Wi-Fi access points to connect users and encrypt their information before it is uploaded to the servers. Only users in spatio-temporal proximity to the uploading user will be able to decrypt and download the information.

Even though every SPICE application will have its own protocol, the underlying framework remains the same. For example, the protocol may differ depending on the type of information that is collected and shared, i.e., an application where a user collects and shares videos and photographs may need to use a protocol different from an application where a user sends information as text. The protocol may also differ in terms of consent management if the photos and videos were taken on the uploading user's smartphone as opposed to being captured by a surveillance camera mounted on a building. In this paper, we present a generic technological framework that allows users to share and receive any information given their spatio-temporal proximity to each other in a secure manner.

Although the broader effects of this idea will need further study via experimental deployment, this paper focuses on the technological

*This work was done primarily when author was a graduate student at Dartmouth College.

CrowdSenSys'17, November 5, 2017, Delft, The Netherlands

ACM ISBN 978-1-4503-5555-1/17/11…$15.00
https://doi.org/10.1145/3139243.3139245

foundations of sharing information in close encounters. We make four contributions:

- We present the design of a smartphone-based system called SPICE (Secure Proximity-based Infrastructure for Close Encounters) to share information in close encounters, while providing strong privacy properties.
- We describe protocols for the SPICE architecture that allows information exchange among users who were in spatial and temporal proximity.
- We further extend the protocol to reduce the risk of unauthorized access to the information and disclosure of users' location history.
- We consider the challenges of a real-world deployment of the SPICE protocol and suggest ways to handle the challenges.

2 THE SPICE SYSTEM

The SPICE system comprises five main components: users, sources, smartphones, broadcasting devices, and servers.

Users. People who have smartphones with wireless capabilities to connect to the Internet and to nearby sources. Although we refer to a user who queries information as a querier q and a user who shares the information as a helper h, over time a given user may be both a querier and a helper. The SPICE system expects that the user carries her smartphone (or tablet) with her at (nearly) all times.

Sources. The helper uses a sensor or a device, i.e., a source s, to capture information. The SPICE system assumes that a user's smartphone is already paired with the sources; the device pairing step is out of scope. For example, photos can be obtained from sources such as smartphone cameras (internal to the smartphone), personal cameras (external to smartphone but with the helper), drones (controlled remotely by the helper), or surveillance cameras mounted on buildings (immovable and not controlled by anyone, but helper can pair with it to obtain photographs).

Smartphones. A user's smartphone stores information collected from sources.

Broadcasting devices. A broadcasting device distributes cryptographic keys to SPICE users in its vicinity. The keys are used to encrypt the information. In this paper, we consider Wi-Fi access points for the role of broadcasting devices.

Server. The server forwards information from helper to the querier. We expect helpers will choose an expiration period for data they upload. By default, all data stored on the server has an expiration period of one month.

2.1 Security Model

We understand that helpers and queriers will have privacy concerns when using SPICE to share and query context. SPICE has the following privacy goals:

P1. *Anonymity.* A matched querier and helper cannot learn each others' identity. The server operator also cannot learn the identity of the querier and helper.

P2. *Unlinkability.* When a querier's smartphone receives information from a helper, neither the querier's smartphone nor the querier can link it to other information provided by the same helper, earlier or later.

P3. *Confidentiality.* Only users who shared a close encounter with the helper can receive information collected by the helper from the source. Also the server does not learn about the users' location history.

We are concerned about the following threats:

T1. *Threat to privacy:* An adversary may link a user to a certain location, link one user to multiple locations, link multiple users at the same location, assemble a complete location history of a user, or identify a user.

T2. *Threat of confidentiality:* A malicious user may try to obtain data collected at locations they never visited.

SPICE has the following trust model.

TR1. *Proper key generation.* The users trust the wireless broadcasting devices to generate keys according to the SPICE protocol and not to share keys otherwise.

TR2. *Proper message routing.* The queriers trust the server to forward their messages to the helpers and the helpers trust the server to forward messages to the queriers.

TR3. *Trustworthy app.* Users trust the SPICE app not to disclose their location history except via the SPICE protocol. Similarly helpers trust the app to not disclose the information they collect except to relevant queriers, and queriers trust the app to send their queries only to the server.

TR4. *Trustworthy sources.* SPICE relies on the sources to determine whether the data they share with users is accurate and to alert the user or administrator managing the sources when error or malfunction is detected.

We expect all access points, smartphones, and sources have sufficient processing power and storage capabilities to support cryptographic primitives. We assume a malicious user cannot install malware on smartphones to collect keys from all users. A user who visits different locations to collect keys is not considered an adversary and can obtain information if they were in spatio-temporal proximity with a helper at those locations, even if their reason for collecting the keys was malicious. We expect a malicious user may be able to collect some, but not all the keys received by all other users, simply by visiting the locations. We assume users will not share keys with other users.

3 DESIGN

We describe the SPICE protocol in three different stages for the purpose of exposition. The terms used in the protocol are defined in Table 1. The protocol has six steps, as described below.

(1) Location tracking: Users obtain location information continuously from Wi-Fi access points; the location information is obtained from beacon frames that are available to all clients in the vicinity of access points and not just the clients that are associated with it.
(2) Information collection: Helpers obtain information from source.
(3) Information upload: Helpers upload information to the server.
(4) Query: Queriers send queries to the server to obtain information.
(5) Consent (optional): The server may send a request to the helper for obtaining consent to share information.
(6) Information download: The querier obtains the requested information from the server.

Notation	Description
u	User
s	Source
h	Helper
q	Querier
L	Location
t	Timestamp
T	Time range
$R(u, T)$	Set of locations visited by u during time range T
B	Broadcasting device
b	Broadcasting interval
S	SPICE server
I, I'	Information
$H(X)$	Message digest of X
$E(k, X)$	Encrypt X using key k
$D(k, X)$	Decrypt X using key k

Table 1: Notations used in the SPICE system model. The type of encryption performed by $E()$ and $D()$ is implicitly symmetric.

Naive version. First, we describe how the protocol allows users to obtain information from other users. For now, we focus on close encounters where both helpers and queriers were at the same location.

Location tracking: The users' smartphones obtain location information from GPS/Wi-Fi access points, i.e., the broadcasting devices.

$$B \Rightarrow u : L, t$$

Information collection: The helpers obtain information from sources.

$$s \Rightarrow h : I$$

Information upload: The helpers upload information tagged with location L, timestamp t to the server S. The server stores the L, t as index along with the information I.

$$h \Rightarrow S : L, t, I$$

Query: Queriers can search for information using a set of search filters, which may include location and a time range.

$$q \Rightarrow S : \{L, T\}$$

Information download: The naive version does not require any consent from the helper. The server retrieves all the information I from the queried location, within the time range T specified in the query and sends the set of all the retrieved information to the querier.

$$S \Rightarrow q : \{I \mid \exists t \in T, (L, t, I) \in S\}$$

In this version, the information is never encrypted, so the users have to trust the server operator to not misuse the data, or disclose the helpers' location history. By routing the communication via the server, the helper and querier never learn each other's identity.

Encrypted version with untrustworthy server. Next, we update the protocol so the helper can encrypt the information before uploading it to the server, so the information cannot be accessed by the server operator. However, the system should allow any querier who shared a close encounter with the helper to access the information, i.e., decrypt the information that was encrypted by the helper. Since the helper and querier shared a close encounter, they were at the same location and so the information can be encrypted using the location where the information was collected. The first two steps remain the same as the naive version.

Information upload: After obtaining the information, the helper encrypts the information using a key computed from an abstract representation of location (obtained by applying a hash function to the location value), and uploads the encrypted information to the server, with the time (in clear text) when the information was taken. The server stores the time as the index of the encrypted information.

$$h \Rightarrow S : t, E(H(L), I)$$

Query: The querier searches for information using a time range T. The queriers do not need to send any location, since the keys effectively abstract their location.

$$q \Rightarrow S : T$$

Information download: The server retrieves all encrypted information that contained timestamps included in the time range requested by the querier. The querier attempts to decrypt the encrypted information I' after retrieving all the locations it visited during the time range T. For those I' records where the querier was at the same location L as the helper, she can decrypt the information.

$$S \Rightarrow q : \{I' \mid \exists t \in T, (t, I') \in S\}$$
$$h : I = D(H(L), I'), \forall L \in R(u, T)$$

In this version, the users need not trust the server operator with the information. The location is never sent in clear text to the server, so the server operator also cannot trace the helper's or querier's location history. But the keys are location values, and hence reproducible; so any user who can guess L will be able to decrypt any information, against the expectation of the helpers.

Encrypted version with malicious users. Finally, we update the protocol to prevent a malicious or curious user from obtaining unauthorized access to information. For this purpose, the keys should be dynamic and reproducible only by someone who visited the same location as the helper at about the same time. SPICE uses forward and backward hash chains to create cryptographic keys that are used for encrypting and decrypting the information, as described below. As additional protection, the server never stores any information; the server forwards any query from queriers to all helpers, so the helpers can give consent before sharing information with the queriers.

The broadcasting devices running the SPICE protocol will broadcast cryptographic keys to all the smartphones (running the SPICE application) in their vicinity, ensuring that any two users at the same location will receive (and can recreate) numbers from the same hash chain. A forward hash chain works as follows. Suppose at time t, broadcasting device B sends key k. At time $t+b$, where the broadcasting interval is b minutes, the broadcasting device sends key $H(k)$, where H is a hash function. At time $t + nb$, B sends key $H^n(k)$. A backward hash chain, on the other hand, pre-computes a different chain of keys and sends them in reverse, as follows. At time t, B sends $H^n(k')$, at time $t + b$, B sends $H^{(n-1)}(k')$ and at time $t + nb$, B sends k'.

Suppose helper h, and queriers q and q', visit location L but never see each other. A querier q who arrived at L before h and leaves at t receives keys $k, H^n(k')$. Helper h arrives at location L at time $t + b$ and receives keys $H(k), H^{(n-1)}(k')$; h leaves before $t + nb$. At time $t + nb$, querier q' receives $H^n(k), k'$. All users store the keys in their smartphones and the keys are not used until a querier requests information. A querier requests the server for any information the system has for a certain time range.

Suppose we were to modify the protocol so that the server stores encrypted information, but in this case, what keys should the helper use to encrypt the information before sending it to the server? The helper only learns the time range for which to generate keys from the query; so in the protocol, the helper never encrypts or sends information to the server, until a query for information arrives. When a server receives a query, the server simply relays this query to all users other than the querier. Alternatively, we could also limit the number of users the server has to forward the query to by having helpers upload the timestamps, to denote when they collected information, which allows the server to choose the helpers to forward the query to. However, we choose privacy over efficiency and to limit the knowledge the server has about users, the helper does not upload any information to the server until a querier sends a query.

Once h receives a query, which contains a time range, she will generate keys that she expects the querier would have received for the time range and encrypts the information with those keys. If the query contains $(t - nb \dots t)$, the helper computes the key using the backward hash function H and encrypts the information with $H^n(k')$. If the query contains $(t \dots t + nb)$, the helper computes the key using the forward hash function H and encrypts the information with all the following keys, $H(k), H^2(k), \dots H^n(k)$. If q requests $(t - nb \dots t + nb)$, h will encrypt I using each of $H^n(k'), H(k), H^2(k), \dots H^n(k)$ as cryptographic keys.

For simplicity, we assume a user only receives keys from one broadcasting device; we discuss multiple broadcasting devices later.

Location tracking: The users obtain cryptographic keys that represent their location from broadcasting devices, such as Wi-Fi access points. At time t, in the above example,

$$B \Rightarrow q : k, H^n(k')$$

At time $t + b$,

$$B \Rightarrow h : H(k), H^{n-1}(k')$$

At time $t + nb$,

$$B \Rightarrow q' : H^n(k), k'$$

There is no information upload step since helpers do not upload any information to the server.

Query: Queriers request information using a time range.

$$q' \Rightarrow S : T, \text{ where } T = \{t \dots t + nb\}$$

Information download: The server does not store any information; it forwards the search to all users, other than the querier.

$$S \Rightarrow u : T, \ \forall u \neq q$$

On receiving the search criteria, the helper's smartphone searches for matching content in its local database, finds all the keys it obtained during the time range and computes all additional keys using the hash chain. The helper generates a one-time key K to encrypt the information and then encrypts the key with all the possible hashes, and sends all the encrypted data to the server. On receiving the message from the server, the querier attempts to decrypt the encrypted key after retrieving all the keys it obtained from locations it visited during the time range T. Once it manages to decrypt the key, it uses the key to decrypt the information.

$$h \Rightarrow S \Rightarrow q' : \{E(K, I), E(H(k), K), E(H^2(k), K),$$
$$\dots E(H^n(k), K) \mid \exists t \in T, (t, I) \in h\}$$
$$q' : K = D(H^n(k), E(H^n(k), K)$$
$$q' : I = D(K, E(K, I))$$

In this version, the users need not trust the server operator or malicious users to refrain from accessing information without authorization. The location is never sent in clear text to the server, so the server operator also cannot trace the helper's or querier's location history. The hash keys can be reproduced, but only by users who were at the location and obtained at least one pair of keys.

4 DEPLOYMENT FACTORS

In this section, we discuss three factors that affect the deployment success of an application that uses the SPICE system. For deployment to be successful, there must be a sufficient number of users willing to share information, so that those who request context may benefit from using the system. Also, there must be a large enough density of Wi-Fi access points to support the mobile users, to ensure the user devices receive cryptographic keys from at least one Wi-Fi access point at all times. Finally, the system must be easy to deploy, i.e., there should be minimal burden on the network administrators to modify the network to run the SPICE protocol.

User Acceptance. Given the popularity of location-based sharing applications (such as Piximity [17], 23Snaps [20], and Path [14]) that allow users to share information in real-time with other users near them, we expect users might also want to retrieve data retrospectively with others who were at the same or similar locations as them at a similar time. Researchers have explored ways to allow users to leverage sensors and resources on nearby smartphones [4], and using SPICE, we allow users to retrieve data retrospectively.

We also expect people might be concerned about using the SPICE system if it discloses their location history, which we address by adding cryptographic keys, or shares information in a manner different from their expectations. The SPICE protocol ensures that no location information is shared with the server or other users, and that only the information chosen by the helper is shared with the querier and then only in encrypted form.

Wi-Fi access points as broadcasting devices. We expect Wi-Fi access points to send keys to all nearby clients within their beacon frames. One factor that affects SPICE success would be the density of broadcasting devices; how often can SPICE users hear a broadcasting device? Achtzehn et al. found a 14-fold increase in Wi-Fi density in urban residential areas over the last decade [1]. The authors revealed a Wi-Fi density of 883, 488, 5179, and 6103 APs per km^2 for the rural residential, industrial, urban retail, and urban residential study areas in Germany, respectively; only 33

access points are required to give full data connectivity, assuming a nominal range for standard 802.11b/g to be 100 meters [6]. Given the high density of wireless access points, we expect that a user will be in the vicinity of multiple access point at any location so SPICE will be able to support its users by using access points. Of course, there may be several locations without any Wi-Fi coverage where users will not see even one access point. SPICE users can still opt to share information collected at this location in a manner described in the first two versions of the protocol, i.e., either without encryption, or by encrypting using the location reading.

Network administrators. We expect SPICE can be easily deployed. It is easy for users to run the application on their smartphones; they do not need to purchase any additional hardware. Also, the protocol does not need to run in the access points; the network administrators can set up a different key server that pushes cryptographic keys through the APs.

5 DISCUSSION

In this section, we consider several issues that may arise in real-world implementations.

Customization. The generic protocol allows helpers to collect information from sources, and upload the details of the information to the server, and queriers to specify the search filters and obtain information shared, with consent, by the helpers. Developers who use the SPICE framework in their crowd-sensing applications can add an additional protocol layer to support different information types and consent structures.

Metadata values. The queriers may want to use metadata values while querying for information. For example, the metadata values for a photograph could be keywords denoting the content of the photograph, the type of camera the photograph was captured with, and the resolution of the photograph. Helpers may also want to upload metadata values to the server, so the server can match a query to their information before contacting them with the query. However, the metadata values may contain sensitive data, and if sent in clear, the server operator will learn sensitive information about the helper and querier. If the application allows users to add metadata values to the information, the application must encrypt the metadata values with the same keys used to encrypt the information.

Multiple APs. A user could be in the range of multiple access points at any time. At time t, a user may be near m access points, and may obtain $2m$ keys.

$$B_1 \Rightarrow h : H(k_1), H^{n-1}(k'_1)$$
$$\dots$$
$$B_m \Rightarrow h : H(k_m), H^{n-1}(k'_m)$$

The keys are stored as a series of tuples on the user's phone, i.e., $(t, H(k_1), H^{n-1}(k'_1)), (t, H(k_m), H^{n-1}(k'_m))$.

To determine whether two users were in a close encounter, we rely on the fact that if two people are spatially close, they may observe the same or similar set of APs; SPICE uses this hypothesis to determine whether the querier was at a location near to the helper's location where the information was collected. Instead of one matching key, the application looks for subsets of matching keys obtained within the same minute t, where the subset size is equal to or greater than a threshold l specified by the application.

Phone as AP. In well-populated regions with limited network coverage, such as remote tourist destinations, it may be possible to use a smartphone as an access point to generate hash chains and transmit keys. In this case, two users will be in a close encounter if they both encountered the same third person at the same location, at about the same time. For efficiency, smartphones can coordinate themselves in agreeing on one immobile phone as a temporary token generator and other moving phones as token receivers.

Hash Key Reset. If the hash chain is never reset, a malicious user will be able to go to different locations, collect keys from different APs, and generate all the keys in the hash chain to obtain all the information collected by helpers. To prevent this scenario, the hash chains must be reset periodically; the hash key reset period is decided by the network administrator. That is, they must be reseeded with a new random key. However, choosing the right time for a reset may be tricky because the helper will not be able to generate keys outside the hash chain. For example, a helper h who collected information right before reset will not be able to share information with a querier q who arrived at the location after the reset. Similarly, if h collected information right after reset, she will not be able to share information with a querier q who arrived at the location before the reset. One option is to perform the hash reset at a time when no helper is likely to collect information, such as 3am.

So far, we assumed in our protocols that any one who visited the location before the hash key was reset can decrypt the helper's information. Even though any user can generate all the keys within a hash chain, information encrypted by the helper should be decrypted only by queriers who were at the location within the time period when the information was relevant, that is, near enough and close enough to be considered a 'close encounter' in the semantics of this application. The time period for which information is relevant depends on the application. In the first two versions of the protocol, the server can easily check whether the information collected by the helper will be relevant during the time range for which the querier requested information and share only if the information is relevant. In the final version, the helper can determine whether any information she collected will be relevant during the time range for which the querier requested information and share only if the information is relevant.

Incentives. We plan to explore credit-based incentive mechanisms to prevent helpers from uploading incorrect information, and queriers from sharing keys for the purpose of malicious downloads and to encourage helpers to share and queriers to request information.

Privacy concerns. We did not evaluate the prototype with real users; users might have additional or different privacy concerns.

A full evaluation of SPICE's usefulness and usability requires deployment with meaningful applications. As future work, we plan to conduct experimental evaluations using a SPICE prototype and a thorough security analysis of the protocols.

6 RELATED WORK

Delay-Tolerant Networks. SPICE is complementary to work on opportunistic delay-tolerant networks that attempt to support peer-to-peer content sharing, even when there is only sporadic connectivity between smartphones [2, 3]; our work differs in that we focus on *close encounters*, which have not been explored in the field of delay tolerant networks.

Location Proofs. There is a lot of prior work on location privacy but we aim to solve a different problem, i.e., how to determine whether users are in a close encounter without exposing their location history. Our work does not track or require proof of the user's exact location, but the SPICE system requires a user to possess or be able to generate the keys to prove they were within spatial range of an access point. One of the first location proof systems, proposed by Waters et al., used round-trip signal propagation latency to prove location [21]. Similar to SPICE, researchers have relied on a user's proximity to an access point [9, 10, 16, 19]; however, unlike the prior work, SPICE requires minimal change to the existing infrastructure. A Privacy-Preserving Location proof Updating System (APPLAUS) allows co-located Bluetooth-enabled mobile devices to mutually generate location proofs, and update to a location proof server [22].

Encounters. SPICE differs from prior work that addresses encounter matching in that SPICE addresses the problem of finding people (and devices) that shared a spatially and temporally close encounter and might have never encountered each other. Our system is similar to *SMILE*, which helps strangers contact each other if they shared an encounter (and can prove to each other that they encountered one another) [11], except SPICE allows strangers to request information from others whom they nearly encountered but did not actually encounter, what we term a "close encounter". *Meetup* supports encounter-based social networks, where co-location is verified by exchanging certificates that contain the user's public key and a photograph [12]; SPICE differs in that one of its fundamental privacy goals is to prevent the helpers and queriers from learning each other's identity. Finally, our work is complementary to techniques used for proximity testing [5, 13, 18]; Narayanan et al. propose private-equality testing of location tags to determine if two users are close by, while our systems allow users to share data based on common keys they would have obtained by being spatially and temporally proximate.

7 SUMMARY

We present a crowdsourcing system that extends the capabilities of location-based applications and allows users to connect and exchange information with other users in spatial and temporal proximity. Our system, SPICE, addresses several privacy issues that arise in location-based sharing, by leveraging Wi-Fi access points to encrypt information before it is exchanged, so only users in spatio-temporal proximity to the helper will be able to decrypt and download the information while the server operator never learns about the information or the users. We present the design of the system, and describe the challenges in implementing the protocol in a real-world application.

REFERENCES

[1] Andreas Achtzehn, Ljiljana Simic, Peter Gronerth, and Petri Mahonen. 2013. Survey of IEEE 802.11 Wi-Fi deployments for deriving the spatial structure of opportunistic networks. In *Personal Indoor and Mobile Radio Communications (PIMRC)*. 2627–2632. https://doi.org/10.1109/PIMRC.2013.6666591

[2] C Caini, P Cornice, R Firrincieli, M Livini, and D Lacamera. 2010. DTN meets smartphones: Future prospects and tests. In *International Symposium on Wireless Pervasive Computing (ISWPC)*. IEEE, 355–360.

[3] H Chenji, A Hassanzadeh, M Won, Y Li, W Zhang, X Yang, R Stoleru, and G Zhou. 2011. *A wireless sensor, adhoc and delay tolerant network system for disaster response.* Technical Report LENSS-09-02. Texas A & M University.

[4] Shane B. Eisenman, Nicholas D. Lane, and Andrew T. Campbell. 2008. Techniques for Improving Opportunistic Sensor Networking Performance. In *Conference on Distributed Computing in Sensor Systems (DCOSS)*. Springer Berlin Heidelberg, 157–175. https://doi.org/10.1007/978-3-540-69170-9_11

[5] L Ertaul, A Balluru, and A Perumalsamy. 2013. Private Proximity Testing For Location Based Services. In *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 1.

[6] Kipp Jones and Ling Liu. 2007. What where wi: An analysis of millions of Wi-Fi access points. In *International Conference on Portable Information Devices (PORTABLE)*. IEEE, 1–4.

[7] Andreas Konstantinidis, Demetrios Zeinalipour-Yazti, Panayiotis Andreou, and George Samaras. 2011. Multi-objective Query Optimization in Smartphone Social Networks. In *Proceedings of the International Conference on Mobile Data Management*. IEEE Computer Society, 27–32. https://doi.org/10.1109/MDM.2011.37

[8] Thomas Ludwig, Christian Reuter, Tim Siebigteroth, and Volkmar Pipek. 2015. CrowdMonitor: Mobile crowd sensing for assessing physical and digital activities of citizens during emergencies. In *Human Factors in Computing Systems*. ACM, 4083–4092.

[9] Wanying Luo and Urs Hengartner. 2010. Proving Your Location Without Giving Up Your Privacy. In *Proceedings of the Workshop on Mobile Computing Systems; Applications (HotMobile)*. ACM, 7–12. https://doi.org/10.1145/1734583.1734586

[10] Wanying Luo and Urs Hengartner. 2010. Veriplace: a privacy-aware location proof architecture. In *Proceedings of the SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*. ACM, 23–32. https://doi.org/10.1145/1869790.1869797

[11] Justin Manweiler, Ryan Scudellari, and Landon P. Cox. 2009. SMILE: Encounter-based Trust for Mobile Social Services. In *Proceedings of the Conference on Computer and Communications Security (CCS)*. ACM, 246–255. https://doi.org/10.1145/1653662.1653692

[12] Abedelaziz Mohaisen, Denis Foo Kune, Eugene Vasserman, Myungsun Kim, and Yongdae Kim. 2013. Secure Encounter-Based Mobile Social Networks: Requirements, Designs, and Tradeoffs. *Transactions on Dependable and Secure Computing* 10, 6 (Nov. 2013), 380–393. https://doi.org/10.1109/TDSC.2013.19

[13] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. 2011. Location Privacy via Private Proximity Testing.. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*.

[14] Path. 2017. Path. (2017). www.path.com

[15] Sameer Patil, Gregory Norcie, Apu Kapadia, and Adam Lee. 2012. "Check out Where I Am!": Location-sharing Motivations, Preferences, and Practices. In *Extended Abstracts on Human Factors in Computing Systems (CHI)*. ACM, 1997–2002. https://doi.org/10.1145/2212776.2223742

[16] Anh Pham, Kévin Huguenin, Igor Bilogrevic, and Jean-Pierre Hubaux. 2014. Secure and Private Proofs for Location-based Activity Summaries in Urban Areas. In *Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. ACM, 751–762. https://doi.org/10.1145/2632048.2632064

[17] Piximity. 2017. Piximity. (2017). http://www.piximity.me

[18] Gokay Saldamli, Richard Chow, Hongxia Jin, and Bart Knijnenburg. 2013. Private Proximity Testing with an Untrusted Server. In *Proceedings of the Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 113–118. https://doi.org/10.1145/2462096.2462115

[19] Stefan Saroiu and Alec Wolman. 2009. Enabling New Mobile Applications with Location Proofs. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (HotMobile)*. ACM, Article 3, 6 pages. https://doi.org/10.1145/1514411.1514414

[20] 23 Snaps. 2017. 23 Snaps. (2017). https://www.23snaps.com

[21] Brent Waters and Ed Felten. 2003. *Secure, private proofs of location.* Technical Report TR-667-03. Princeton University.

[22] Z. Zhu and G. Cao. 2011. APPLAUS: A Privacy-Preserving Location Proof Updating System for location-based services. In *International Conference on Computer Communications (INFOCOM)*. 1889–1897. https://doi.org/10.1109/INFCOM.2011.5934991

[23] X. Zuo, A. Chin, X. Fan, B. Xu, D. Hong, Y. Wang, and X. Wang. 2012. Connecting People at a Conference: A Study of Influence between Offline and Online Using a Mobile Social Application. In *International Conference on Green Computing and Communications*. 277–284. https://doi.org/10.1109/GreenCom.2012.52