Secure Encounter-based Mobile Social Networks: Requirements, Designs, and Tradeoffs

Abedelaziz Mohaien, *Student Member, IEEE,* Denis Foo Kune, *Student Member, IEEE,* Eugene Vasserman, *Member, IEEE,* Myungsun Kim, and Yongdae Kim, *Member, IEEE*

Abstract—Encounter-based social networks and encounter-based systems link users who share a location at the same time, as opposed to the traditional social network paradigm of linking users who have an offline friendship. This new approach presents challenges that are fundamentally different from those tackled by previous social network designs. In this paper, we explore the functional and security requirements for these new systems, such as availability, security, and privacy, and present several design options for building secure encounter-based social networks. To highlight these challenges we examine one recently proposed encounter-based social network design and compare it to a set of idealized security and functionality requirements. We show that it is vulnerable to several attacks, including impersonation, collusion, and privacy breaching, even though it was designed specifically for security. Mindful of the possible pitfalls, we construct a flexible framework for secure encounter-based social networks, which can be used to construct networks that offer different security, privacy, and availability guarantees. We describe two example constructions derived from this framework, and consider each in terms of the ideal requirements. Some of our new designs fulfill more requirements in terms of system security, reliability, and privacy than previous work. We also evaluate real-world performance of one of our designs by implementing a proof-of-concept iPhone application called MeetUp. Experiments highlight the potential of our system and hint at the deployability of our designs on a large scale.

Index Terms—Social networks, Location-based services, Privacy.

1 Introduction

In the conventional model of social networks, users select their contacts from a set of off-line acquaintances. Despite their utility, these conventional networks support only a subset of social networking: two users will only be able to establish a relationship in the social network if they know of, or are introduced to each other. On the other hand, in an encounter-based social network, the only requirement for establishing a connection is to be in the same place at the same time—similar to striking up a conversation at a public place. Encounter-based social networks would provide a computing infrastructure to allow for creation of varied services such as a "missed connections" virtual bulletin board, on-the-fly introductions (business card exchange), or real-time in-person key distribution to bootstrap secure communication in other systems.

Although at first glance encounter-based systems appear very similar to existing social networks, they present a dramatically different set of challenges, not the least of which are security and privacy of users and authenticity of the other party in a conversation. Guarantees that are trivial in traditional social networks, such as authenticity (ensuring one is communicating with the desired person), become open problems in encounter-based networks. Additionally, requirements like anonymity—a feature that is not needed in most traditional online social networks based on prior face-to-face contact—need to be considered in encounter-based networks. This is desirable because users would expect information about people they happen to meet to stay private. Furthermore, since people do not automatically place their trust in others simply based on presence in the same location, it is also desirable to *reveal the minimum amount of information required* for future secure communication. Sharing detailed personal information is not the primary goal of encounter-based networks, but can of course be easily implemented if both users agree upon the successful verified encounter.

In this paper we consider fundamental requirements for encounter-based social networks. We note that in addition to basic functionality like high availability, scalability, and robustness to failure, these systems should provide several security guarantees, including privacy in the form of unlinkability of users sharing an encounter, confidentiality of data exchanged among encounter participants, and authentication of both users in a two-party conversation. We show that SMILE [27], a recent state-of-the-art design, fails to meet a number of these requirements (even though it was built explicitly with security in mind). We propose a generic design that can be used to construct networks that provide different security guarantees. We then describe individual designs and show the benefits and trade-offs of specific security design decisions.

- *A. Mohaisen is with Verisign Labs, Reston, VA 20151, USA. Email: amohaisen@verisign.com*
- *D. Foo Kune is with the Computer Science Department, University of Massachusetts, Amherst, MA 01003, USA.*
- *E. Y. Vasserman is with the Department of Computing and Information Sciences, Kansas State University, Manhattan, KS, 66506.*
- *M. Kim is with the Information Security Engineering Department, University of Suwon, Suwon, South Korea.*
- *Y. Kim is with the Electrical Engineering Department, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea.*

Unlike prior work, we provide fine-grained separation between the encounter event and the eventual connection and communication: authentication and communication may happen immediately, or may be delayed for an arbitrary period of time. The former provides unlinkability between the two paired users (a third party cannot determine that two users have made a connection), while the latter increases convenience and flexibility at the cost of somewhat degraded unlinkability. However, both schemes guarantee authentication—that once established, the connection is with the desired user. Both of these designs consist of an "online phase," where the encounter takes place and encounter instance information is exchanged, and an "offline" or delayed communication phase, where encounter information is used for the two parties to reconnect and communicate privately. It is worth noting that *we assume that other users at the encounter time and location are potentially malicious*, and may collect information, collude with other parties, and otherwise make it difficult for two people to establish a secure private connection.We developed a prototype of our design, called MeetUp[1], that uses visual authentication for encounter information exchange and verification. At the core of our system is a visual authentication scheme that provides authenticity guarantees for users involved in an encounter. Our authentication scheme capitalizes on that people are good at remembering faces but worse at remembering names Encounter-based networks with visual authentication would play to people's strengths, allowing anyone who remembers a face to later connect with the "owner" of that face, without the need to remember additional information. MeetUp uses Tor hidden services [14] to provide an anonymous communication channel for the second phase of our protocol. By performing preliminary real-world experiments using plausible deployment settings, and considering user feedback, we highlight the end-user usability of our system and its feasibility for deployment at larger scales.

While the main contribution of this paper is an encounter-based social network design, our techniques can be employed for a wide range of applications, such as a drop-in replacement for a face-to-face key distribution service for future secure communication, e.g. SPATE [23], or for privacy-preserving file sharing systems, e.g. OneSwarm [20]. In OneSwarm, untrusted users get their keys from an online key distribution center. Using our design, one may distribute keys to untrusted users based on some shared activity—an encounter. Any application that requires key pre-distribution, such as storage services, private file-sharing systems, private collaboration groups, etc, would benefit from our design in the same way. Another example is a scientific meeting, where some researchers present their work, and others participate in discussions, and no one has time to introduce themselves to everyone. We can employ our encounter-based system for private on-the-fly name and business card distribution—concerte examples are discussed in §6.4.

Our contributions in this work are as follows. (*i*) by first outlining security and functional requirements that are ideally desired for encounter-based social network and arguing that these are minimal requirements for many distributed system with reasonable security and privacy guarantees, we examine the extent to which SMILE, a recent state-of-the-art design of secure encounter-based social network, meets these requirements, showing that it is vulnerable to many attacks. (*ii*) we propose a new and generic architecture for encounter-based social networking that greatly differs from the architecture of previously proposed systems and suggest two possible implementations, each striking a balance between performance and security. (*iii*) we show the feasibility of our designs by implementing a proof-of-concept system—including an iPhone application called MeetUp—conforming to our requirements and evaluating its performance in real-world settings using mobile devices, and by bringing further evidence on the usability of our design and rationality of used assumptions based on several user studies.

The organization of this work is as follows. In §2 we describe idealized security and functional requirements expected in encounter-based networks. In §3 we discuss some of the related work in the literature, followed by a discussion of vulnerabilities of SMILE. In §4 we introduce the design of generic encounter-based social network and discuss two specific designs. In §5 we discuss the implementation of MeetUp, and details of some of the experiments that we performed to illustrate the usability of our design. In §6 we highlight the main discussion points followed by concluding remarks in §7.

2 Requirements and Challenges

As we have mentioned in §1, many encounter-based designs do not consider even basic security and privacy requirements along with functionality and performance. Others fail to meet these requirements even though they were created with the explicit goal of satisfying them. Below, we explore some requirements for idealized secure encounter-based social networks. While this list is by no means complete, it can be used as a preliminary guide for evaluating past and future designs.

2.1 Security Requirements

Here we outline some of the desired security features of encounter-based social networks. Note that these requirements are generic in the sense that they may apply to *many distributed systems* which combine human interaction, sensitive private information, and network communication. The security requirements we expect in these systems are as follows. (*i*) *privacy* or *unlinkability*. The privacy of two parties sharing an encounter must be protected, even from others in the vicinity who may also participate in simultaneous encounters. In this case, privacy means that an external adversary (even one taking part in the encounter or colluding with a "bulletin board" or rendezvous server to be used in latter phase) who is not one of the two users of interest should not be able to conclusively determine that two users have made a connection. (*ii*) *authenticity,* meaning that when two users decide to make a connection, they should be assured that messages indeed originate from each other. (*iii*) *confidentiality*, meaning that information exchanged between two users should be accessible only to them.

1. http://www.cs.umn.edu/~foo/meetup/

2.2 Functional Requirements

The following are generic functional requirements in the context of large-scale distributed systems that are also desirable for an encounter-based social network. (*i*) *availability*. As such, the infrastructure to exchange encounter information should be accessible *most of the time*. The unavailability of individual users should not affect the availability of other users. Since the time at which encounter parties check for potential encounters associated with their activities could be arbitrary, the encounter-based social network is more sensitive to availability than conventional social networks. (*ii*) *scalability*. With typical social networks being large in size, any potential social network design, including those based on encounters, should scale to support a large number of simultaneous users. This requires minimizing dependence on a centralized entity (our rendezvous server mentioned above).

3 Background and Related Work

While it may appear that implementing these requirements would be straightforward, it is surprisingly challenging in practice. Recently, Manweiler et al. devised SMILE [27] to implement a subset of these requirements. While they succeed in meeting some of the functional requirements, their system does not protect against a number of common security vulnerabilities, such as the "man-in-the-middle" (or MitM) attack, which leads to several other breaches as shown below.

Closely related to our work, as well as to SMILE, are GAnGS [9] and SPATE [23], which are both systems built to facilitate secure data exchange among groups in an authentic manner using simple human factor techniques. GAnGS extends demonstrative identification (DI) to a group setting. The original allows users to indicate which two devices should communicate at a time, and is by nature designed for pairwise grouping. The basic idea of GAnGS is to use device pairing in an efficient manner for groups by using auxiliary tools such as projectors for inputting information about the group (GAnGS-P) or by depending on other users in the group to perform tree-based pairing (GAnGS-T). Unlike our work, while GAnGS can be used for encounter-based attestation, it is mainly designed for collaborative data authentication. SPATE [23] improves on GAnGS by streamlining cryptographic operations to make the system more usable on mobile devices. Neither work considers privacy or anonymity of participants, since authentication and collaboration are done at the same phase, and any potential attacker can maintain participants in both designs easily by eavesdropping on communication taking place between them. Related to both works, although with slightly different applications, is SafeSlinger [17]. SafeSlinger emphasize usability when creating trust among participants in communication and on-the-fly collaboration settings.

Since location is one of the most frequently used pieces of information for encounter verification, location proofs are studied in [32] and [21]. Some commercial platforms that utilize the idea of short-range communication and location-based services include Brightkite [7] and Loopt [24] while other similar ideas can be seen in WhozThat [5], Serendipity [16], SocialAware [18], Veneta [34], D-book [10], and Bump [8] (an application for contact information exchange that provides no privacy guarantees against a compromised central server; its security is analyzed and improved in [33]), among many others.[2] Most of these works do not consider location privacy, despite of its importance.

Last system worth mentioning is MobiClique [30], which builds an ad-hoc on-the-fly mobile social network by bootstrapping initial contacts from online (static) social network. As in our work, users in MobiClique use short-range Bluetooth communication and can establish encounter-based social links. Most interestingly, MobiClique provides several measurements demonstrating the feasibility of such system in terms of power consumption on typical mobile devices, as the one used in our design. However, unlike our system, MobiClique does not guarantee user privacy.

► **Overview of SMILE.** The main work in the literature that is similar to our work in goals and purpose is SMILE. SMILE extends ideas from [26] to establish trust between individuals who shared an encounter. It attempts to allow users equipped with mobile devices to build such trust relationships while preserving their privacy against potential attackers (e.g., the rendezvous server and other users in the encounter settings). In SMILE, users who want to communicate with each other must prove that an encounter occurred between them. To do this, the first device in the encounter generates and broadcasts the "encounter key" to other devices within its communication range. The same device then posts a cryptographically-secure hash of the encounter key, along with a message encrypted using the encounter key to a centralized server. Due to the pre-image resistance properties of the hash function, the centralized server cannot recover the encounter key without help, and thus cannot read the message. Other users of SMILE with the same encounter key may claim the encounter by looking up the hash of the key, which is used for indexing the encrypted message at the centralized server. Only users with the correct key will be able to decrypt the message left by the first encounter party at the server, and every user with the correct key can derive the retrieval hash value. The benefits of the basic design of SMILE as it is described here is that it reduces the misuse in the encounter system: only people who have been at the encounter place are those who know the encounter credentials and are able to claim the encounter.

In addition to the basic design, SMILE tries to provide two features: k-anonymity and decentralization. k-anonymity is achieved by truncating the hash values of the keys so that a single user is concealed amongst k other users with the same truncated value. SMILE features a decentralized system that uses anonymizing networks of re-mailers for communication, claiming to provide k-anonymity by requiring each user to have at least k identifiers.

► **Requirements met by SMILE.** We now examine which of the previously-derived requirements SMILE meets. The system's availability and scalability are limited, *since the system depends on a centralized server that is easy to disrupt—a problem that is not unique to* SMILE*, but rather any*

2. Note that some of these applications give the impression of short-range communication but actually communicate over the Internet.

design that uses a centralized online entity. Additionally, the claimed security guarantees might not meet the requirements outlined above. While the confidentiality of encounter-related information is safeguarded by encryption, the privacy of users in SMILE can be breached. In principle, while the problem exists in systems that rely on a centralized server, one can augment the performance of SMILE and mitigate the problem by providing a server with high availability guarantees, which comes at cost that need to be considered as part of the design.

First, SMILE is prone to an *impersonation attack* performed by a user present during the encounter. Since no authentication is done during key agreement, any user can eavesdrop on the encounter information and later claim to be the party of interest. This attack can further be extended to monitoring: if the adversary exchanges keys with the first user pretending to be the second, and repeats this with the other user, the adversary can carry out a MitM attack and monitor all messages passed between users.

Second, SMILE is prone to *user collusion*, an attack that was previously reported in social interactions [25]; a few malicious users colluding with the rendezvous server may possess enough information about activities of other honest users (such as timestamps, locations information, and encounter keys) for the server to unmask users, determining the identities of communicating parties.

Finally, while not particularly a specific problem of SMILE but every system using such a building block, the k-anonymity in SMILE requires that each user know the number of other nearby SMILE users in order to make sure that there are enough people around to mask the activity of an individual—that the user is indistinguishable from k others in a given encounter setting. This, however, can be easily misrepresented by a Sybil attack [15] where a single adversary pretends to be $k-1$ other SMILE users, *compromising honest user's anonymity*.

► **A comparison with SMILE.** While we take a different approach than that used by SMILE, and any comparison between our approach and that of SMILE might turn unfair, we conclude this section with the main differences in guarantees and functionality of our design and that of SMILE. First of all, our design is scalable by nature, particular when considering the design option of using hidden services where users run their own servers for post-encounter communications. Second, our design provides stronger authentication features, by visual means, thus preventing the MitM attack, whereas SMILE does not provide these features. On the down side, the use of visual means for authentication is not universally accepted—see our user studies in section 6— and might have a privacy cost associated with it, while SMILE does not use such feature although at the risk of enabling the MitM attack. Finally, our designs, centralized or decentralized, provides better guarantees for the post-encounter phase by using a mix network to access encounter information, thus reducing the risk of giving away additional information to the potential adversary (impersonator or centralized server) by concealing networking information of users. While this feature can be added easily to SMILE, the fact that weaker authentication is used in it would still enable a variety of attacks the adversary can perform—e.g., collusion. In our design, impersonation is hard to launch, since it requires colluding with the centralized signing authority, while SMILE does not make use of such authority, having the advantage of being lighter-weight than our design, but enabling the collusion in its attack surface.

4 Designs and Design Options

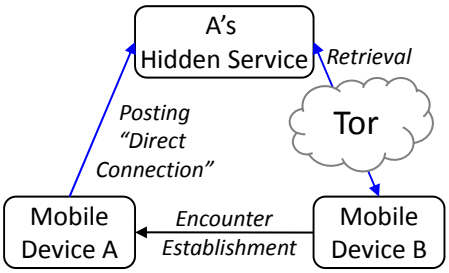
With requirements outlined earlier, we generalize the design of previous systems. Special attention has been given to the security and privacy requirements previous designs failed to achieve. We divide the design into functional blocks and describe potential attacks on various parts of the system. Then, we discuss two instantiations of the generic design; each with different benefits and trade-offs.

4.1 Functional Components

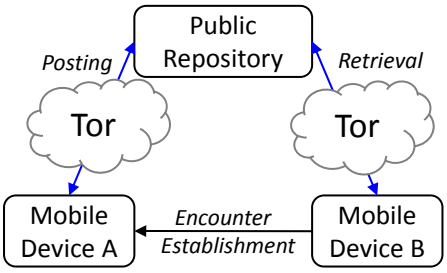
The functional design of a typical encounter-based social network consists of three major components located at three different architectural layers, shown in Fig. 1: the user layer, the plug-in layer, and the "cloud." The term cloud may refer to a storage location of the encounters and private messages (e.g. a central rendezvous server or distributed "mini-servers") which is used by different encounter parties in the post-encounter phase. However, the design can be quite flexible, allowing storage components to be dynamically chosen using a plug-in architecture: the system may support centralized servers, distributed hash tables [28], or even Tor hidden services [14]. Notice that each of the different layers provides functionalities used to realize one or more functional or security requirement among these explained in §2. Furthermore, to establish a balance between the functional and security requirements, we also discuss two specific designs in the next subsection. Below, we elaborate on what requirements each designs meet.

4.1.1 On the Need for Strong Authentication

We have shown in section 3 that simple unauthenticated key agreement during the encounter is vulnerable to a man-in-the-middle attack. Given that the parties involved in the encounter are already aware of each other visually, the only way to avoid this vulnerability is to enforce a visual authentication scheme where users can recognize that they are communicating with the desired party simply by looking at a picture of that user. In other settings such as a professional conference, a company logo and other information, which could be viewed as a reduced digital version of a business card (though, in many cases, the same scenario of using a personal photo on a personal business card still applies). To provide user authentication, we assume each user to have a digital certificate signed by a trusted authority with sufficient information to identify users, including a photo of the user. The signing authority's public key would be known to all other nodes who use our encounter-based social network. It is not far-fetched to assume that future authentication tokens such as passports and driver licenses will be issued digitally, since cryptographic signatures make them more secure against malicious tampering than their physical counterparts. Though, we don't use such certificate but a limited one (see details in below). With that assumption,



(a) Hidden service-based design



(b) Repository-based design

Fig. 1. Two specific designs. Fig. (a) illustrates the first design using Tor hidden services as encounter storage place. Fig. (b) illustrates the second design where users store encounter information on a public replica and gain anonymity to their access using a normal Tor operation.

a user of an encounter-based system broadcasts a certificate with his or her picture and public key which is received by other people in the encounter space including the intended destination. Such information is then used for reconnection according to one of the design options explained in §4.2.

We note that facial recognition algorithms exist, which might reduce the privacy of the user, when an attacker collects photos from certificates being exchanged and compare them to photos associated with names and obtained from other sources such as other online social networks [2]. Although these attacks are computationally expensive, one may argue that the use of cheap cloud services may make these attacks very feasible. However, we answer this concern by pointing out three issues. First, even when using such cloud service, the attack, unless targeted towards a particular user, would be infeasible with a substantial cost that the attacker has to pay in order to breach the privacy of users who use our system. Second, the attacker does not need to collect broadcast certificates in order to apply the attack, but may simply take pictures of the encounter space and achieve a similar result to prove the presence of an individual at a certain place at a certain time. Finally, all prior work of facial recognition depends greatly on features extracted from original photos, but not from cartoon versions of them, which could be used to remedy the privacy breach associated with using a photo for visual authentication. [3] Our user study that considers cartoon version of photos instead of the original photos indeed hints on improved usability of our design.

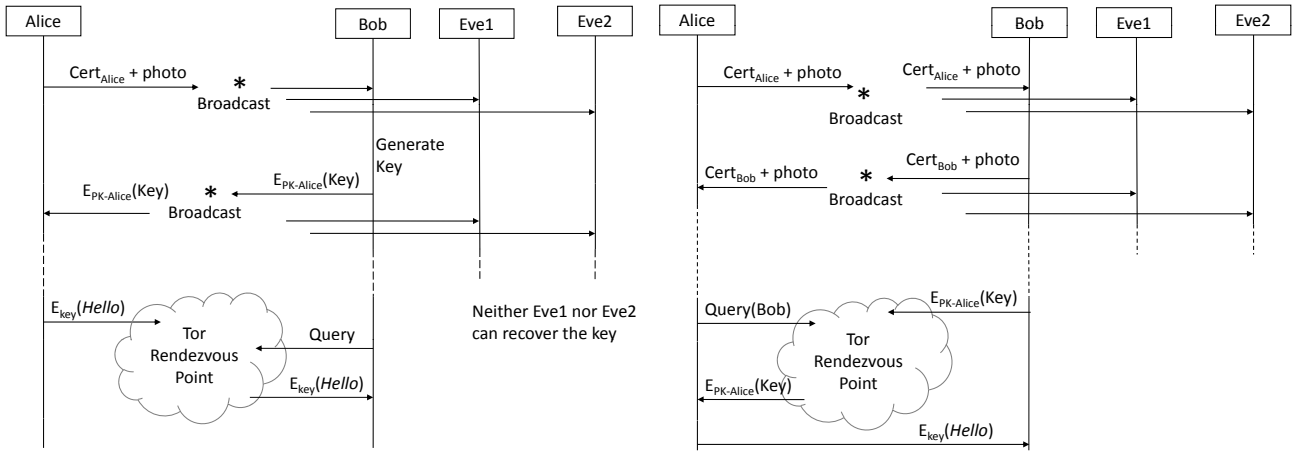
3. Indeed, one possible remedy to this attack is to use cartoon version photos, like the widely accepted and used photos in StreetPass and Mii.

4.1.2 Trusted Certification

In our design, we use the X.509 standard [12] for certification without any modification to the structure of the certificate, but we limit the attributes available in the certificate used for encounters (discussed below) in order to preserve the privacy of our users. Indeed, the X.509 standard allows optional attributes for biometric information such as photos, which enables us to embed visual information into the certificate. The trusted authority mentioned previously is responsible for ensuring that the photo provided by user for certification is an actual representative picture, and allows others to visually identify the user. So, even when issuing a certificate that combines multiple pieces of private information, such as the certificate owner name, address, etc.), the authority will issue a separate, limited certificate with reduced amounts of private information which fits our social encounters design (only user's public key and photo). The ultimate signature by the trusted authority will sign all embedded attributes in the certificate, including the photo. Notice that the centralized authority used for signing the certificate with the photo does not need to be online for the protocol to work. Indeed, after the initialization phase of the protocol, in which certificates are issued for the participating parties in our design, verification of the signatures embedded in the certificate are verified at the side of the receiving party of the certificates using a publicly known public key of the authority. On the other hand, this guarantee comes at a cost that is not used in SMILE. Indeed, SMILE is lightweight since it does not require a certification entity, yet the certification entity provides stronger guarantees for the authenticity of participants. More details on the rationale of using such entity are provided below.

Our certification and visual authentication schemes are very simple. First, a user Alice generates a pair of public and secret keys (PK, SK), computes the hash value of her own image and other relevant information, including a Tor hidden service URI, which is a unique identifier that is used later by Bob to communicate with Alice over Tor hidden service. Alice embeds her PK and other metadata into a certificate request, and sends it to a signing authority. Second, the signing authority checks the validity of the metadata hashes in the certificate request and verifies the validity of the used attributes in relation with the previously mentioned extended certificate. If the verification process is successful, the signing authority signs the certificate using its own private key and sends it back to Alice. If at any time through the verification process any of the above conditions do not hold, the signing authority aborts and refuses to sign. Notice that here we omit some critical details: the authority only signs the certificate with the photo only if correctness of the photo associated with the physical identity of Alice can be established, e.g. by physical presence of Alice at the authority.

At the protocol's run time, Alice broadcasts her certificate to everyone in the vicinity, along with the photo as credentials, which will later allow anyone present in the encounter space claim an encounter with Alice and proceed to the next phase depending on the protocol being used. In one of such design options — let Bob be one of the people to overhear the



(a) Immediate key exchange

(b) Delayed key exchange

Fig. 2. Sequence diagrams of our encounter-based social network design with two key exchange scenarios: (a) shows the immediate key exchange with postponed authentication and encounter reconnection via the Tor network, while (b) shows delayed key exchange and delayed rendezvous via Tor network (hidden service or direct Tor connections).

broadcast, if the photo passes the visual authentication by Bob, Bob tries to verify if the certificate along with the photo are genuine, i.e. have a valid signature from a trusted authority. Bob computes the hash of the photo and other information sent by Alice, comparing it to the value embedded in the certificate. Assuming a match, Bob proceeds to verify the signature on the certificate by using the public key of the signing authority. If the signature is valid, then Bob admits Alice to be whoever she claims. Otherwise, Bob aborts. Notice that this authentication process can be deferred to post-encounter phase, as it is the case in delayed rendezvous.

4.2 Design Options

In the generic schemes outlined in §4.1 we face two potential choices: do we require an immediate encounter key agreement between the two parties, or do we wait? Each approach has a benefit and drawback. Immediate generation of an encounter key requires manual selection of the target user while still at the encounter point. Delayed generation, on the other hand, requires no immediate action on the part of the user, but potentially erodes user privacy during later communication. Both of these methods are discussed further below. Note that these are not options to be selected within a single system; this choice must be made before deployment to have a consistent protocol among all users in the network.

4.2.1 Immediate Pairing

If a user is willing to manually select the picture of other users of interest while still at the encounter site, she can compose an encounter key, encrypt it to the selected user's public key, and broadcast the resulting message. Each user in the vicinity will detect the transmission and attempt to decrypt it. However, only the target user will be able to decrypt the message correctly, and thus recover the encounter key. This key will be used later to exchange private messages at the rendezvous point. This method prevents the rendezvous server and colluding adversaries from determining which two users are communicating. We can go a step further and use timed-release encryption [31] to hide the contents of the message even from its intended recipient until the encounter is over, ensuring that users do not inadvertently give themselves away by using their devices at the same time. In principle, time-release encryption would allow A sequence diagram showing the operation of this key generation design is in Fig. 2(a).

While the advantage of this design option is enabling users to make decisions while at the encounter space—while they remember well parties they encountered, enabling direct communication, and utilization of the physical encounter, reasoning about some security guarantees in this scenario might not be as easy. Particularly, unconventional attacker capable of measuring signal strength, and associating that to users might be able to breach the privacy of users by matching who meets whom by monitoring the encrypted traffic between them, thus violating the unlinkability requirement.

4.2.2 Delayed Rendezvous

Devices will consistently broadcast their certificates, but will not require others users to immediately review the information. (As in the immediate pairing scheme, we can use timed-release encryption [31] to enforce this constraint.) At a later time, the device user can look at the list of collected identities (and public keys) and select those with whom he wishes to communicate. As before, we will use non-malleable encryption to compose a message to the other user, but now the message must be stored "in the cloud" in such a way that it is linkable to the public key of the user for whom it is intended, and some encounter nonce passed at the time of the encounter. This may not be a significant problem, considering that only keys and faces are exposed, and not more personal components of users' identities. A sequence diagram showing the operation of the two key generation design options is shown in Fig. 2(b). While this scheme does not suffer from the shortcomings in the immediate pairing scheme, the capability of reconnecting to encounter parties depends entirely on the capability of encounter parties to recall such encounters. We

believe remembering people is quite easy, given the limited number of encounters per time window.

4.2.3 Decentralization and Anonymity

Our distributed design, one that does not require a rendezvous server, is depicted in Fig. 1(a). We use the generic design described in §4.1 combined with Tor hidden services [14] to provide communication anonymity. While Tor provides users with anonymity, Tor hidden services enable servers to conceal their identities as well. Each user runs his own Tor hidden service and uses it for two purposes: first, to hide his identity and gain anonymity as to his location and second, to serve follow-up requests relating to previously encounters. The other party must use the Tor client to access the hidden service, also gaining anonymity and hiding her location from the server. This design can easily scale to a large number of simultaneous users [22], and is resilient to failure, since an attack on the entire social network built using this distributed design would require attacking many individual nodes simultaneously (i.e. the failure of one hidden service would not affect other hidden services).

4.2.4 Centralized Design with Anonymity Guarantees

Our second design is depicted in Fig. 1(b). Here we assume a public repository to which users involved in the encounter can post encounter information. Suppose that Alice shares a public space with Bob, and therefore learns his public key from his certificate. At an arbitrary time after Alice and Bob share a location, Alice can go through all her collected identities, notice Bob's picture, and decide to strike up a conversation. She composes a message to Bob, encrypts it under Bob's public key, and posts the encrypted message on the centralized repository under Bob's public key. To gain anonymity as to her identity and location, Alice uses a Tor client, concealing her IP address from the central server. This is more efficient than the hidden services used in the previous protocol, which require one of the encounter parties to be online all the time to serve other parties involved in the encounter. In this design, on the other hand, Bob can get the messages left for him at the central repository at any time. He similarly accesses the repository through Tor to conceal his identity, and downloads all messages addressed to him. To identify such messages, we suggest using nonces as part of the indexing scheme. These random one-time values, generated and exchanged at runtime of the encounter protocol, along with the public key of the encounter party that initiates the encounter, are hashed and used for indexing. By doing so, a malicious repository will not be able to get any information about the identity of the person accessing the repository unless at least one person at the encounter site is malicious and colluding with the repository. Notice that the use of Tor here is not to preserve the privacy of the participants from an adversary present at the encounter, but from the storage server himself. The use of Tor conceals the participants IP address, and thus location, and disable breaching privacy by associating participants with such information. It is noted, however, that such collusion with the server is not possible when using Tor's hidden services, since each participant acts as his own server; although this option comes at some cost by requiring participants to be online

5 Implementation and Experiments

To validate our method and assess the practicality of our design, we implemented the system on the iPhone platform and tested it on multiple devices under ideal conditions, as well as conditions that users are likely to encounter in urban settings. In our implementation, we used the delayed rendezvous scheme where the user's device can collect simulated broadcast information during encounters and then use the decentralized Tor hidden service architecture for the second part of the encounter. Those require a hidden service URI (an address through which one can access services deployed by hidden servers [14]) to be part of the user's information and is thus linked with the certificate as a bundle in sent the transmissions.

Notice that, even when an adversary captures the certificate exchanged between two honest participants, and get access to the URI, the honest participant running the hidden service will still have a full control over whether to respond to requests for communication sent via the hidden service. Accordingly, while the use of the hidden service would resolve the rendezvous problem and provide means for reconnection in the future based on the previous encounter, it will increase the attack surface by enabling means for the adversary to breach the privacy of the users and their encounter.

Notice that our design is generic. We are not limited to any specific platform like Apple's iOS, which we chose for development, in any of the our design ingredients. Our choice of development platform for our proof-of-concept application is only due to availability and ease of use for quick prototyping. Other platforms, such as Android, would work just as well. Consequently, any conclusions on the usability of our design are independent of the platform, as we only require a smart phone with basic wireless capabilities.

5.1 MeetUp: An iPhone Application

Our iPhone application, called "MeetUp," allows users to find other users of the system within Bluetooth range, decide with whom they wish to communicate, and send and receive private messages. Screenshots of typical usage scenarios are shown in Figures 3(a) through 3(c). The user searches for other nearby users of our system, and receives their identification information, including photographs and certificates signed by our trusted certificate authority.

5.1.1 Certification and Visual Authentication

The certificate authority uses a scaled-down version of the architecture presented in section 4. Certificates signed by the authority include hashes of photos and Tor hidden service URI unique to the user. The file containing the certificate, the photo, the hidden service URI, and the signature are the deployed to each device in the system. The certificate authority is responsible for verifying that only one instance of such file is deployed per user. It is also responsible for verifying that the photo matches the user. This is similar to machine-readable biometric authentication used in modern passports [13]. A

larger deployment of our system could rely on an already-implemented certificate infrastructures that use photographs, such as a driver's license records, as discussed in §4.1.

5.1.2 Wireless Communication

Inter-device communication was implemented using Bluetooth [6]. The limited range of Bluetooth devices ensures that users are within close physical proximity to exchange certificates. This makes it more likely that users are within visual range and can identify each other. For the delayed key exchange, we rely on that humans can easily recognize a face that has been seen before [19] when we present multiple devices that have been observed previously, along with photos relevant to the owners.

Our next step was to implement the broadcast protocol over Bluetooth. Unfortunately, the Bluetooth specification does not explicitly support broadcast in the way we require. One broadcast scheme allowed by the Bluetooth standard is one over a piconet [6] in which a small number of devices within radio range can form a temporary ad-hoc network. Broadcast communications then take place over those small networks. Unfortunately, the Apple SDK does not support piconets in iOS 4.1 [4] at the time of writing of this paper. The only option on the iPhone platform was to use Bluetooth peer communication, using repeated unicast to emulate broadcast. It has a major drawback since both parties in a peer session will have to acknowledge each other's devices before any information exchanges can take place. This will obviously cause problems for the delayed message setting, but in our case it was sufficient to obtain some RF measurements data for our experiments. Fortunately, since we strictly emulate broadcast, this forced implementation choice does not violate our security guarantees. Bluetooth sniffers and strength indicators might be able to help in localizing a transmitting device, but that information is already assumed to be public. As for the recipient, if a user connects to every other device in the vicinity one by one in random order and exchanges equal amounts of data with all of them, an adversary cannot determine the intended recipient.

Notice that the Android platform enables broadcast [1], and a potential final product of our design could be realized on the Android platform—although the main determining factor for using the iOS platform at the time of writing this work was the availability of iOS devices to develop the application and to test the design. However, it is noted that the alteration of the platform will have less impact on the validity of the rest of the results, particularly the user studies, since recent studies have shown that the Android OS is taking dominance as it is used on 52% of the smart phones in the US, as opposed to 35% for the iOS [11]. A deployment on an Android device would require a single message for a single broadcast, instead of the simulated scenario described above which is necessitated for demonstrating the basic idea with existing equipments.

5.2 Using MeetUp

The first step in using our application is for the user to start scanning for devices within Bluetooth range. The applications

(a) Searching (b) Users found (c) User details

Fig. 3. iPhone App implementation screenshots

on remote phones have to be listening as well to start a Bluetooth peer session. Once the session is established, the two devices can exchange certificates, photos, and signatures. With this method, however, the initiating device has to go through the list of nearby devices and do an individual pairing with each remote device in turn instead of doing a real broadcast, adding considerable overhead.

In the following measurements, we considered the time required to transfer a 20KB bundle between two paired devices. We discounted the pairing time since it was a step necessary only to our emulated network. In a real broadcast, there will be no pairing time. We did not have access to the proper equipment (such as a spectrum analyzer) to actually measure the amount of traffic on the 2.4GHz band, so we chose locations with minimal RF interference and densely populated areas around the campus with a heavily utilized 2.4GHz band for urban setting.

5.2.1 Effective Range

We used an open field in a sparsely populated area to obtain ideal condition measurements. Such an environment ensures minimal interference over the 2.4GHz Bluetooth communication band and minimal multi-path due to signal reflecting off of objects around the communicating devices. Our experiments indicate that under those conditions, the devices can discover each other and exchange information at a range up to 24 meters. Transfer times increased as we increased the distance between the two devices, but all were faster than 400ms (shown in Figure 4). We also looked at the directionality of the communication to determine if users have to be pointing their devices in a particular direction to ensure timely transfer of information. We measured the time required to transfer 20KB of data over our Bluetooth channel from a user holding a device in a particular manner. Measurements were taken at 45° increment by a querying device moving around a responding device. The experiment was repeated for radii of 1, 2, 3 and 4 meters around the responding device. We did not find any significant transfer time differences for all of our measurements. The median transfer time was approximately 250ms for the 20KB payload (measurements are shown in Figure 5).

5.2.2 Effective Range with Obstacles

We consider the time taken to transfer and receive encounter information between two encounter devices under several conditions reflecting real-world deployment settings, where

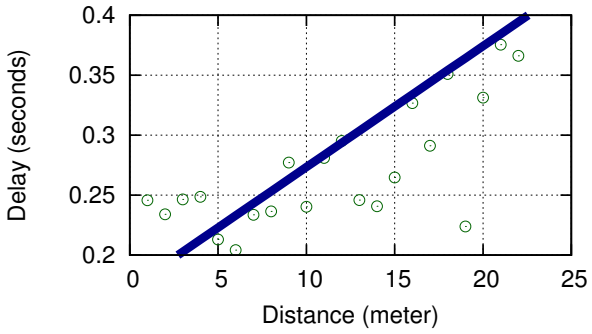


Fig. 4. Delay as the time it takes to send encounter information (about 20KB) and receive it by other encounter parties with variable distance without obstacles.

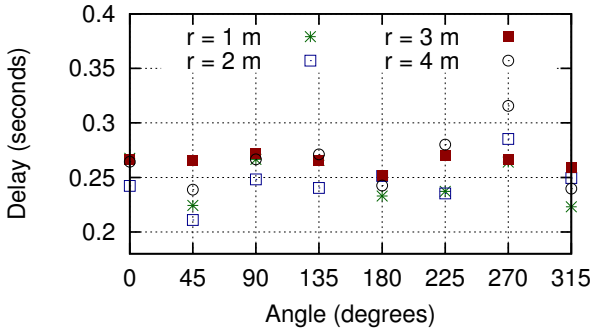


Fig. 5. Delay with different locations of the encounter sender and receiver, as determined by a radius r (in meters) and an angle θ (0 to 325 with 45 degrees increment).

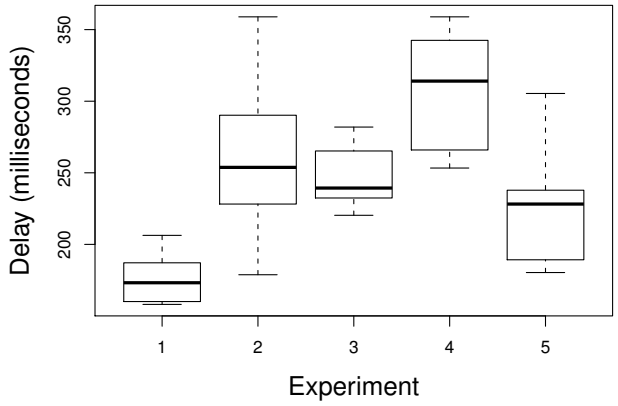


Fig. 6. Delay with several scenarios representing different potential deployment settings of MeetUp.

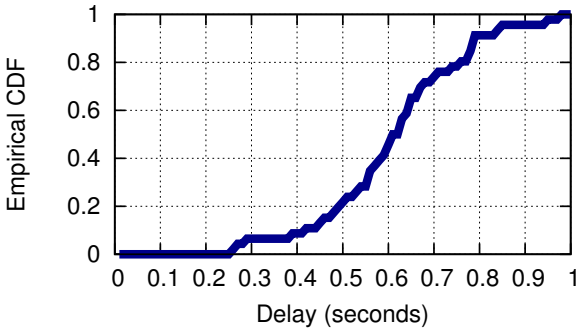


Fig. 7. Delay as the time it takes to send encounter information (certificate and data in a 20KB bundle) and receive it by other encounter parties in urban settings

obstacles around the encounter parties may cause signal attenuation and multi-path interference. We consider five communication scenarios of interest: (*i*) through a barrier (door), (*ii*) in hallway—line of sight with a separation of 20 meters, (*iii*) communication across multiple walls, (*iv*) while on different floors (2 floors separation), and (*v*) when one of the parties is in an elevator and the other is outside it. For each scenario, ten measurements are taken and the results are shown in Figure 6, where we plot the 5 representative values of min, max, median, Q_1 and Q_3 (median of the first and second halves of the measurements). While some scenarios imposed far greater delay than others, the data generally shows feasibility of MeetUp in several potential deployment settings.

5.2.3 Measurements in Urban Settings

We tested MeetUp in a densely-populated urban setting, in a bus station populated by students equipped with mobile phones, with this being as the only difference from the range and obstacles experiments above. The data collected from this experiment are shown in Figure 7. We observe that it takes less than a second in all cases to do the encounter, and at average it takes approximately 600 ms. While larger than an environment free of obstacles, multi-path and interference, the transfers were still completed in an acceptable time window, thus supporting the practicality of our design. For this experiment, only two users participated in the experiments where one is the initiator of the encounter and the other is the receiver. The results shown in Figure 7 correspond to roughly 100 readings.

5.2.4 Tor Hidden Service

Following the device encounter and data transfer over the wireless network, we used a Tor hidden service for the second phase of the anonymous encounter. We transfered a 40KB data bundle that only the intended recipient will be able to decrypt. We established a new Tor circuit for each experiments, and we ran multiple timing measurements per experiment. The timings tend to be very consistent per circuit but very different between circuits (ranging from 1.5 to about 8.5 seconds). Most circuits we used showed an acceptable transfer delay of under 10 seconds. Notice that the Tor hidden service does not increase the attack surface, but rather hides the users' additional network information, such as IP address, while enabling the rendezvous in a decentralized fashion.

5.2.5 Technical Issues

During our tests we noticed that a number of users tend to use their real names as their device's names due to the default naming scheme used by the mobile device. The device name (which in many cases contains the user's name) is transmitted to other devices during the Apple iOS Bluetooth

pairing protocol. Such a naming scheme would defeat the purpose of the anonymity provided by the protocols outlined in this paper. In future implementations we intend to use the time of encounter as a selector, instead of a device nickname. From a user's point of view, the appropriate photos will be displayed next to the time of encounter allowing proper selection. The list can also be augmented with locations at which the encounter occurred.

In our implementation, we exported the certificate from the device to a desktop which then created a rendezvous point over Tor. There is also an option to have the device itself connect to the Tor network to set up a rendezvous point. At the time of writing this paper, the iPhone Tor client required a jailbroken iPhone, limiting its utility. Depending on the policies of the iTunes store, we may be able to include a Tor client component within our application, or rely on a connection to a desktop computer to establish the rendezvous point and wait for incoming connections. Other future implementations of MeetUp would consider other potential mobile devices that already support Tor, e.g., Android platforms [3]. Notice that if one is to consider the iPhone platform to serve as the platform of MeetUp, one only needs to give up the option of running hidden services over the iPhone while other options are, including running hidden services on a desktop, would be still available and are independent of the platform.

6 Evaluation and Discussion

6.1 Privacy Evaluation

6.1.1 Privacy in the Encounter Phase

In the first phase of the encounter (when users are still in the same location), the first party—referred to as the encounter source—uses a broadcast communication channel that makes the second party of the encounter—which we refer to as the encounter destination—unlinkable to the source. Information broadcast by the source is received by every other party in the encounter space, and no destination information is revealed. The only information revealed about the source is her public key and photo. We discuss the privacy implications of this setup below. However, it is clear that while an adversary present at the encounter can determine who else is present and using MeetUp, the adversary cannot determine if any two users made a connection. In the post-encounter phase and for the purpose of reconnecting with users who were present during an encounter, the identity and location of the person initiating the connection to the rendezvous server or hidden service are obscured using the Tor network, or Tor hidden service, though in the latter case is immediately revealed to the source of the encounter if verified.

6.1.2 Privacy in the Post-Encounter Phase

While it is easy to reason about the second case where a user runs his own hidden service, since the security of the communication is inherited from that of the Tor network and computing entities under the full control of the user, it is more difficult to determine whether unlinkability holds when using a centralized rendezvous server. Since encounter information is deposited on the central server by the destination and is based on the source's information (e.g. an index derived from his public key), this information might be used to breach the privacy of users — any entity may check the source's mailbox to see if there is a message. Note that the message is encrypted, and therefore still confidential. The problem is alleviated using encounter-time random nonces, which would be combined with the destination party identity to derive the rendezvous key used by the encounter parties.

Only a malicious server colluding with a user in the encounter space would be able to extract any information from the rendezvous key — the server acting alone gets no useful information. Furthermore, the extracted information is limited to the time and place of the encounter, and nothing else. The certificate of the source, if the destination decides to repost it as an evidence of the encounter, is encrypted under the source's public key along with the message left to the source at the central server. Such information about the source cannot be linked to any other information, since the server in our designs does not store such information, unlike the case of SMILE. Notice that another reason for using per-encounter nonces generated at encounter time is to optimize the communication overhead when retrieving the encounter information. A fix for the colluding server and malicious users mentioned above is to make each user who wants to retrieve encounter information posted to him on the server to perform "dummy" queries to disguise his query.

6.1.3 Privacy Concerns due Visual Authentication

One may criticize our design for using a personal photo associated with the encounter information, which may be eavesdropped by all users in the encounter setting, including the attacker. While the photograph-to-key binding may be abused to degrade the privacy of users, we argue that this is a necessary piece of information, and a potential attacker might learn it from several other sources, apart from this application. We further argue that such information is already available to the attacker by physically co-locating with the encounter party, and by seeing who is present at the same place in the same location. However, we stress that this information cannot be used to breach the privacy guarantees of the encounter, since the adversary cannot read messages exchanged between users, nor does he know the identity of the other party in the encounter. We finally argue that users interested in maintaining unlinkability provided in our design for their encounters are also willing to give this piece of private information away, for the ultimate benefits gained. This claim is further supported by the following user study.

We emphasize again that our scheme requires a visual authentication to avoid certain security breaches, namely the MitM attack, while SMILE does not (at some security cost). Our goal is to ensure that we are indeed sending messages to the appropriate party. At the point of a short encounter, the users do not have any information about the recipient, except for a visual information, which is driving our requirement for visual authentication. This requirement does have a privacy angle to it, but in the age of public facebook profile pictures, users are finding it more acceptable to have photos

of themselves available, which is—as pointed out earlier—already available through other sources. This includes, as an analogues case to the technique used in our work, having a video camera running during the train ride and capture a photo of everyone. More details on the usability associated with this building component are highlighted in the following.

► **Survey settings.** To demonstrate the usability of our design, we perform several user studies as shown in the subsequent paragraphs. In these user studies we recruited 76 volunteers with age ranging from 21 to 35 through a Facebook application designed especially for this study. Invitations to participate in the survey are made to the sample size from a larger pool of candidates, of approximately 320 subjects, and each candidate in the sample is selected from the pool uniformly at random. The average age of the respondents to the study was 25.7 years old, with 45 of the respondents being males while the rest being females. Of the responding subjects, 68 had at least a bachelor's degree (or were enrolled in a program that leads to a bachelor's degree at the time of the survey; about half of them are enrolled in or had a graduate degree) while the rest had a two years or less of education post high school. Now we proceed to describe these user studies in more details.

► **User Study 1: Using Photos for Authentication.** To understand the potential of our design in real contexts, we perform a case study on a random sample of 76 subjects described above and examined their willingness to use their personal photos as part of an authentication method in order to improve their privacy. Out of the 76 subjects, 4 subjects did not respond (correspond to 5.26% of the sample size). Thirty six (36) subjects responded positively by agreeing to use their photos (correspond to 50% of respondents and 47.37% of the overall sample size) while 22 responded negatively (correspond to 30.6% of the respondents and 28.95% of the sample size) and 14 (18.42% of the respondents and 19.4% of the sample size) selected to turn the feature on at times. In total, 50 out of 74 respondents (correspond to 69.4%) are likely to use the feature by providing their personal photos for certification and authentication when using the service.

We repeat this user study, but at this time by asking the same subjects whether they would be willing to use our application if their photos were to be replaced by a cartoon version as a remedy to privacy concerns. Out of the subjects who negatively responded in the earlier study, 8 indicated it is likely to turn on the application at times, while 4 indicated their willingness to use the application. 10 (correspond to 13% of the population) responded negatively. In total, 87% are likely to use the feature by providing their personal photo or a cartoon version of it.

6.2 Overhead and scalability

The overhead required in MeetUp is in the form of communication, computation, and memory. Communication resources are required for transferring and receiving encounter information, computations are required for establishing Tor circuits, in normal and hidden-service based operation, and memory is required for storing the encounter information in the mobile device and later on a desktop machine that is used for running the hidden service. While both are considered for the resources requirements, of interest to our feasibility study is the mobile device used for carrying out the encounter operations. Here, we verify the feasibility of MeetUp and its reasonable consumption of resources.

As we have shown in the previous section, the time it takes to exchange encounter information in our design is small, and in most cases is less than 1 second on typical devices. Furthermore, in many of the deployment environments that we have considered, this overhead is even about 250 milliseconds, making it very feasible to use.

The memory required in our design, per encounter, and shown earlier in §5, is about 20KB. While this is large in relation with previous memory consumption requirements for similar designs, such as SMILE, we believe that this is reasonable for the provided guarantees, and given the amount of resources in many of the current smart phones which are equipped with GBs of memory. For example, with a 512MB allocated for the application, one may store up to more than 25,000 encounters. Given that one has the choice to decide to store the encounter or discard it right away, this space of memory can be further utilized to store more useful encounters. Also, given that the offline communication and reconnection is performed through non-mobile machines, as suggested by our design, this memory can be further utilized in a better way: the memory required on the mobile phone is only for fresh encounters, which are limited per days [27]. On a desktop machine, 1GB of memory is enough for storing 50,000 encounters per user, far more than the number of friends one can realistically have.

The computations in our design are mostly cheap to perform on typical mobile devices. The only online computations required in our design is a signature verification in order to verify the authenticity of certificates issued by the certificate authority. This overhead can be further minimized by considering verification for encounters that pass the visual authentication, or can be further moved to non-platform in an offline phase. This decision, however, may or may not be desirable based on the traseoff set by users between computations and memory consumption (as one needs to store all encounters, including undesirable ones, in order to perform verification offline). In total, the computation required in our design is reasonable and feasible for most mobile devices.

On the other hand, offline computations and communication required for our application are different from those required on the mobile phone. While memory requirements are still same, in the offline phase we use Tor, or Tor hidden services, to provide privacy. By measuring that for the same amount of communication overhead (20KB), we found the time it takes to transfer such information over Tor (using previously established circuit) is about 3 seconds. This further supports the feasibility claims of our design.

One may argue against the usability of MeetUp, given typical smart phones limited batteries which may drain quickly due to the heavy use of Bluetooth communication. However, we observe that even when one keeps MeetUp running all time and scan for encounters every two minutes, typical smart phone battery would serve for more than eight hours, as it is shown in [30] with MobiClique, in which the overhead is comparable to the overhead in MeetUp.

6.3 Usability Issues

Our design assumes the availability of smart phones for users and their willingness to use their phones to participate in the system. To understand the density of smart phones and willingness of people to use them in our application, we perform the following user study.

► **User Study 2: On Using Smart Phones.** We examine the survey outcome on whether subjects are willing to use their smart phones for applications such as MeetUp or not. In the same sample, 25% of the questioned subjects did not respond, implying the likelihood of not having smart phones or not willing to use their phones for social networks for such applications as MeetUp. However, 39.47% of the sample (52.6% of the respondents) answered positively, 28.95% (38.6% of the respondents) answered negatively, and 6.58% (8.8% of respondents) answered with "maybe" for the likelihood of using their smart phones to connect with people they meet. Out of 57 respondents, 35 (correspond to 61.4%) are likely to use smart phones for connecting to people they meet in MeetUp.

► **User Study 3: On the Density of Wireless Gadgets.** The typical use case for the MeetUp application is in a social setting where people congregate. To this end, we chose a library on the campus of a major North American university. Since MeetUp runs on the Apple iOS platform, we design an experiment to estimate the density of devices capable of running our software in the chosen location. Notice that this user study does not test whether these devices' owners are willing to use our application or not, since this is already tested in the two previous user studies. Accordingly, the majority of the devices did not have our application installed. Given the results of our two previous case studies, one can infer that the presence of such population of gadgets at anytime in the suggested deployment scenario would make a sound conclusion on the usability of our application.

Apple builds a service within their devices that help in zero configuration situations. For this service, the devices use the name assigned by the user for communication between devices on a local network. By default, that name contains the type of device it is. 'The first step in this protocol is a Multicast DNS query on the local network to check for name collisions. The devices run this protocol by default upon first joining any Wi-Fi network.

We connected a laptop to the local Wi-Fi network at our chosen location and listened for Multicast DNS messages. To ensure that we were limited to the target location, we had a user with a known device name connect to the same network, but at different location, and verified our inability to observe his device's DNS messages. We estimated the area of the target location to be around 5000m^2.

We collected messages heard on the Wi-Fi network for 5 hours, and filtered the Multicast DNS queries. We then extracted the unique IEEE MAC addresses of the querying device from those messages, and eliminated any duplicates. We verified that all the MAC addresses belong to the IEEE OUI prefixes assigned to Apple to filter out any devices with an iOS name, without being one. Those default self-assigned names identifies the device type as "iPhone", "iPod" and "iPad", allowing us to estimate the iOS device diversity.

To enable a social encounter with mobile devices, it is important to for those devices to be at the same location at the same time. Observing Multicast DNS messages tells us when an iOS device joins the network, but we don't know how long it stays. We can make a rough estimate based on the physical properties of the location. With the radius of our location being 35m and the average human walking speed of 1.4m/s, we estimate that a user will stay in the network for at least 25 seconds. Using the Multicast DNS messages, we can estimate a lower bound for the number of devices coming online at the above time intervals. In our results shown in figure 8, we counted the number of devices announcing on the network within 25 second buckets. We filtered out duplicate DNS requests, and replies originating from the querier within the reply timeout window to avoid double counting devices. On average, we observed about 9 devices joining the network every 25 seconds. The measurement started around 1pm local time, which would explain the initial bump in devices, followed by a gradual decline into what would be dinner time locally.

Limitations of our counting technique include the following. (*i*) the iPhones have to be configured to connect to the university's Wi-Fi network. While not counting all devices, this is a plausible case since the Wi-Fi data network is much faster than the 3G network on campus, therefore users have an incentive to turn Wi-Fi on. (*ii*) the names of the devices can be changed to remove the device type. We don't know the fraction of user who would do so, but we have at least a lower bound on the number of Apple iOS devices

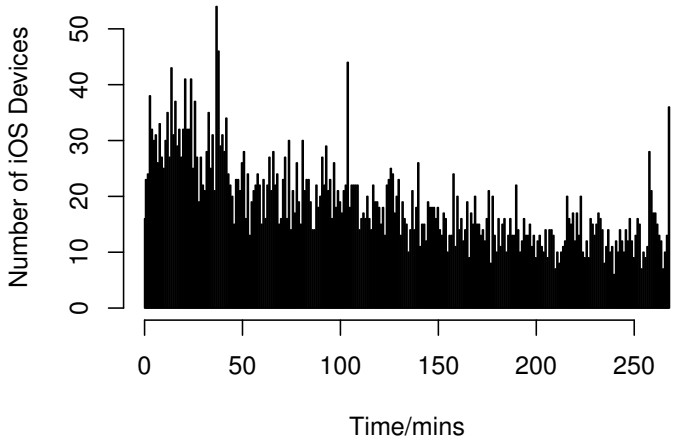


Fig. 8. Apple iOS device density estimation

Even with those limitations, we were able to observe 448 unique devices, including 257 iPhones, 129 iPods and 62 iPads on the network within 5 hours in an area of 5000 m^2. Measuring the Multicast DNS messages indicated their presence on the network, possibly at different points in time as the device sleeps and wakes up while staying in the same geographic location. This density of devices provides us with some confidence that the MeetUpapplication could be useful in augmenting a social network graph based on geographically

proximate social encounters.

6.4 Additional Applications

Now we turn our attention to further applications. We elaborate on applications—mentioned in §1—that may require anonymity for shared encounters. Among many others, we discuss two examples.

6.4.1 Key Distribution

Key distribution is a challenging problem in the context of distributed computing systems. One obstacle for key distribution is the fact that it is hard to make an authority always online to take care of the distribution of keys, as well as the scalability issue of key distribution for larger networks. Our proposed design can be utilized for key distribution, and can be used as a plug-and-play service for this purpose. For example, consider the application of OneSwarm in [20]. In OneSwarm, there are two classes of users, trusted and untrusted users, and both are used for different purposes and differ in the way they get keys and their function in OneSwarm. While the trusted users get their keys from those who trust them directly in an "offline" fashion, untrusted users get their keys from a key distribution center, that should be online all the time. Using our design, one may distribute keys to untrusted users based on activity shared with them—such as an encounter. One even may consider the scenario of establishing trust based on the encounters [23]. Other key distribution applications that may benefit from our design include storage services, file-sharing, etc.

6.4.2 On-the-fly Name Card Distribution

Consider the scenario of scientific meeting, where some researchers present their work, some others participate in discussions on the work, and none has time to keep in touch and introduce himself to all researchers, due to the time constraints. Our application can be brought in action for such scenario for on-the-fly name or business card distribution. Again, same as the main motivation of our application, people are good at remembering faces of other encounter people rather than names, and so it is easy to associate a digital name card associated with a photo than that of remembering names.

7 Conclusion

In this work we show that existing designs for secure encounter-based social networks fail to fulfill reasonable security guarantees. We outline several requirements that ideal encounter-based social networks need to satisfy, and introduce a generic framework for constructing encounter-based social networks. We then use our framework to showcase several designs, and demonstrate that our designs fulfill more requirements than SMILE, the design the motivates our work. We show the feasibility of our work through a demonstration of MeetUp, an iPhone application that uses our design. In the future, we will investigate further extensions to the current framework, alternative designs options, and additional pluggable components. We will also investigate developing MeetUp on other mobile platforms as well as a larger-scale deployment using multiple wireless communication protocols.

Acknowledgement

We would like to thank Max Schuchard for his valuable feedback on an earlier version of this work. An earlier version of this work appeared in [29].

References

[1] Android Broadcast Documentation. http://goo.gl/FTxzV.

[2] A. Acquisti, R. Gross, and F. Stutzman. Faces of facebook: Privacy in the age of augmented reality. In *BlackHat*, 2011.

[3] Android development kit. http://developer.android.com, October 2010.

[4] Apple Inc. Apple iOS Networking & Internet. http://developer.apple.com/technologies/ios/networking.html, October 2010.

[5] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han. Whozthat? evolving an ecosystem for context-aware mobile social networks. *IEEE Network*, 22(4):50–55, 2008.

[6] Bluetooth. Bluetooth Specification Version 4.0. *Bluetooth SIG*, 2010.

[7] Brightkite. http://brightkite.com/, October 2010.

[8] Bump. iPhone and Android application. bu.mp/, 10 2010.

[9] C.-H. O. Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang, and T.-C. Wu. GAnGS: gather, authenticate 'n group securely. In *MOBICOM*, pages 92–103, 2008.

[10] R. J. Clark, E. Zasoski, J. Olson, M. H. Ammar, and E. W. Zegura. D-book: a mobile social networking application for delay tolerant networks. In *Challenged Networks*, pages 113–116, 2008.

[11] CMS Wire. Android dominates burgeoning us smartphone market. http://goo.gl/WZ4tZ, August 2012.

[12] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (IETF RFC 5280). Internet Engineering Task Force, Request For Comments, 2008.

[13] COUNCIL REGULATION (EC) No 2252/2004 of 13 December 2004 on standards for security features and biometrics in passports and travel documents issued by Member States. Official Journal of the European Union, December 2004.

[14] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the USENIX Security Symposium*, 2004.

[15] J. Douceur. The sybil attack. *P2P Systems*, pages 251–260, 2002.

[16] N. Eagle and A. Pentland. Social serendipity: Mobilizing social software. *IEEE Pervasive Computing*, 4(2):28–34, 2005.

[17] M. Farb, M. Burman, G. Chandok, J. McCune, and A. Perrig. Safeslinger: An easy-to-use and secure approach for human trust establishment. Technical Report CMU-CyLab-11-021, Carnegie Mellon University, 2011.

[18] C. M. Gartrell, W. C. M. Gartrell, D. S. Mishra, S. Charles M. (m., and C. Science. Socialaware: Context-aware multimedia presentation via mobile social networks, 2008.

[19] P. Hancock, A. Burton, and V. Bruce. Face processing: Human perception and principal components analysis. *Memory and Cognition*, 24:26–40, 1996.

[20] T. Isdal, M. Piatek, A. Krishnamurthy, and T. E. Anderson. Privacy-preserving P2P data sharing with OneSwarm. In *SIGCOMM*, pages 111–122, 2010.

[21] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi. Location-based trust for mobile user-generated content: applications, challenges and implementations. In *HotMobile '08: Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 60–64, New York, NY, USA, 2008. ACM.

[22] J. Lenhard, K. Loesing, and G. Wirtz. Performance measurements of Tor hidden services in low-bandwidth access networks. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *ACNS*, volume 5536 of *Lecture Notes in Computer Science*, pages 324–341, 2009.

[23] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun, and B.-Y. Yang. SPATE: small-group PKI-less authenticated trust establishment. In *MobiSys*, pages 1–14, 2009.

[24] Loopt. http://loopt.com/, October 2010.

[25] M. Macy. Learning to cooperate: Stochastic and tacit collusion in social exchange. *The American Journal of Sociology*, 97(3):808–843, 1991.

[26] J. Manweiler, R. Scudellari, Z. Cancio, and L. P. Cox. We saw each other on the subway: secure, anonymous proximity-based missed connections. In *HotMobile '09: Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, pages 1–6, New York, NY, USA, 2009. ACM.

[27] J. Manweiler, R. Scudellari, and L. P. Cox. SMILE: encounter-based trust for mobile social services. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *ACM Conference on Computer and Communications Security*, pages 246–255. ACM, 2009.

[28] P. Maymounkov and D. Mazières. A peer-to-peer information system based on the XOR metric. In . I. P. of the 1st International Workshop on Peer-to Peer Systems (IPTPS02), editor, *IPTPS*, 2002.

[29] A. Mohaisen, E. Y. Vasserman, M. Schuchard, D. F. Kune, and Y. Kim. Secure encounter-based social networks: requirements, challenges, and designs. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 717–719. ACM, 2010.

[30] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. MobiClique: middleware for mobile social networking. In *WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks*, pages 49–54, New York, NY, USA, 2009. ACM.

[31] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, MIT, Cambridge, MA, USA, 1996.

[32] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *HotMobile '09: Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, pages 1–6, New York, NY, USA, 2009. ACM.

[33] A. Studer, T. Passaro, and L. Bauer. Don't bump, shake on it: the exploitation of a popular accelerometer-based smart phone exchange and its secure replacement. In R. H. Zakon, J. P. McDermott, and M. E. Locasto, editors, *ACSAC*, pages 333–342. ACM, 2011.

[34] M. von Arb, M. Bader, M. K. 0002, and R. Wattenhofer. Veneta: Serverless friend-of-friend detection in mobile social networking. In *WiMob*, pages 184–189, 2008.

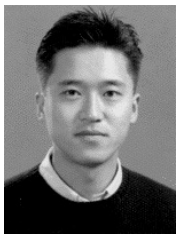
Abedelaziz Mohaisen obtained his M.S. and Ph.D. degrees in Computer Science from the University of Minnesota, both in 2012.

In 2012, he joined Verisign Labs where he is currently a Research Scientist. Before pursuing graduate studies at Minnesota, he was a Member of Engineering Staff at the Electronics and Telecommunication Research Institute, a large research and development institute in South Korea. His research interests are in the areas of networked systems, systems security, data privacy, and measurements.

Denis Foo Kune is a postdoctoral research associate at the University of Massachusetts Amherst where he is focusing on defending against vulnerabilities in the wireless medium. His work includes short-range electromagnetic compatibility, embedded devices and cellular networks. He has over a decade of experience in the industry, mostly at the Honeywell Labs where he worked on improving wireless protocols, industrial networks, and represented Honeywell on technical standard committees. Denis received his MS and PhD in Computer Science from the University of Minnesota, and has a B.A. in Computer Science from Macalester College.

Eugene Vasserman is an Assistant Professor of Computing and Information Sciences at Kansas State University. He received his Ph.D. and Master's degrees in Computer Science in 2010 and 2008, respectively, from the University of Minnesota. His B.S. in Biochemistry and Neuroscience is also from the University of Minnesota (2003). He is interested in distributed network security, privacy, anonymity, censorship resistance, mobile and pervasive computing, and usable security.

Myungsun Kim received the B.S. degree in computer science and engineering from Sogang University, Seoul, Korea, in 1994 and the M.S. degree in computer science and engineering from the Information and Communications University (ICU), Daejeon, in 2002. He received the Ph.D. degree in mathematics from Seoul National University (SNU), Seoul, in 2012. Currently, he is an assistant professor in Department of Information Security, University of Suwon. He was with the Digital Media Research and Development Center, Samsung Electronics, until 2008. His research interests include multiparty computation in cryptography.



Yongdae Kim is a professor in the Department of Electrical Engineering at KAIST. He received PhD degree from the computer science department at the University of Southern California under the guidance of Gene Tsudik. Prior to join KAIST, he was a faculty member in the Department of Computer Science and Engineering at the University of Minnesota - Twin Cities. He received NSF career award and McKnight Land-Grant Professorship Award from University of Minnesota in 2005. Currently, he is serving as a steering committee member of NDSS (Network and Distributed System Security Symposium). His current research interests include security issues in various systems such as cyber physical systems, mobile/ad hoc/sensor/cellular networks, social networks, storage systems, and anonymous communication systems.