enClosure: Group Communication via Encounter Closures

Lillian Tsai*
MIT

Roberta De Viti
MPI-SWS

Matthew Lentz
University of Maryland

Stefan Saroiu
Microsoft Research

Bobby Bhattacharjee
University of Maryland

Peter Druschel
MPI-SWS

ABSTRACT

New applications enabled by personal smart devices and the Internet-of-Things (IoT) require communication in the context of periods of spatial co-location. Examples of this encounter-based communication (EbC) include social exchange among individuals who shared an experience, and interaction among personal and IoT devices that provide location-based services. Existing EbC systems are limited to communication among participants that share a direct encounter.

This paper is inspired by two insights: (1) encounters also enable group communication among devices connected by *paths in the encounter graph* that is contextual, spontaneous, secure, and does not require users to reveal identifying or linkable information; and (2) addressing communication partners using encounter closures subject to causal, spatial, and temporal constraints enables powerful new forms of group communication.

We present the design of *enClosure*, a service providing group communication based on encounter closures for mobile and IoT applications, and a prototype implementation for Android and the Microsoft Embedded Social Cloud platform. Using real-world traces, we show that enClosure provides a privacy-preserving, secure platform for a wide range of group communication applications ranging from connecting attendees of a large event and virtual guest books to disseminating health risk warnings, lost-and-found, and tracing missing persons.

CCS CONCEPTS

• **Security and privacy** → *Pseudonymity, anonymity and untraceability*; *Privacy-preserving protocols*; *Social network security and privacy*; • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools.*

KEYWORDS

mobile computing, privacy, group communication, encounter-based communication, Internet-of-Things (IoT)

ACM Reference Format:
Lillian Tsai, Roberta De Viti, Matthew Lentz, Stefan Saroiu, Bobby Bhattacharjee, and Peter Druschel. 2019. enClosure: Group Communication via Encounter Closures. In *The 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19), June 17–21, 2019, Seoul, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3307334.3326101

*This research was performed at MPI-SWS on a Fulbright grant.

MobiSys '19, June 17–21, 2019, Seoul, Republic of Korea

ACM ISBN 978-1-4503-6661-8/19/06.
https://doi.org/10.1145/3307334.3326101

1 INTRODUCTION

Personal smart devices and stationary IoT devices enable many context-aware applications and services. For instance, people may interact—via their smart devices—with nearby people and resources, or follow up with others they have encountered, directly or indirectly, in the past. Users may wish to follow up with other attendees after a meeting; share commentary about an event they attended or a place they visited; identify people with shared interests in their proximity; confirm that they have met at a given place and time; and use local resources or control their present physical environment.

Prior work has shown that *encounter-based communication (EbC)* [5, 15, 27, 29, 31] can meet these communication needs while providing strong properties: (i) EbC enables communication during and after an encounter without the need to pair devices or exchange contact details; (ii) EbC is *privacy-preserving* because it does not require users to disclose their identity, whereabouts, or communication to a third party; (iii) EbC enables communication partners to authenticate as devices encountered at a given time and place; (iv) EbC ensures message confidentiality and integrity. With EbC, any pair of devices within range of Bluetooth forms a *secure encounter*, consisting of a shared secret E_s, a unique public identifier E_{id}, and the time and location of the encounter. Using an untrusted Cloud-based key-value store, devices may communicate securely with any party they have previously encountered, using E_{id} as the key and encrypting messages with E_s.

Previous EbC systems have been limited to communication among devices that have directly encountered each other. Our work in this paper is based on two key insights. First, secure encounters can also enable communication among parties connected by paths in the graph of encounters, while retaining the same strong properties as direct EbC. Second, addressing communication partners using closures of the encounter graph subject to causal, spatial, and temporal constraints enables powerful new forms of group communication. We explore the properties, opportunities and challenges associated with *group communication via encounter closures*, and present and evaluate enClosure, a library service for Android and Microsoft Embedded Social.

enClosure enables rich, powerful, contextually meaningful communication among devices and users who have never directly met. For instance, enClosure enables communication among users who occupied the same large space at the same time but never met; users who visited the same place at different times; users who crossed each others' trajectories at different times; and users connected by causal chains of encounters that may imply a flow of information or spread of disease.

As we will show, enClosure facilitates communication among users who shared an experience like a large event or journey; communication among users and stationary devices in their wider vicinity; virtual guest book and context-based recommendation services; lost-and-found services; targeted dissemination of health risk warnings; and even aid in the investigation of missing person cases. Users can retain the opportunity to participate in these forms of communication in a privacy-preserving manner, i.e., without disclosing their identity, whereabouts, or linkable information, merely by running the enClosure service in the background.

enClosure differs fundamentally from existing forms of group communication. Unlike in conventional address-based group communication, users need not exchange individual addresses or agree on a group address. Unlike in publish-subscribe services, users need not agree on a topic ontology or reveal their interests. Unlike in Web-based location services like Foursquare, Facebook Places, or Google Latitude, users need not disclose their identities, whereabouts, social connections, and communication to a third party. With enClosure, users address intended receivers via encounter closures with causal, temporal, and spatial constraints. This form of addressing gives enClosure unique expressive power to name communication partners, without requiring users to reveal upfront their interests, contact details, or other linkable information.

In the remainder of this paper, we discuss related work and applications of enClosure; present the design of enClosure and an analysis of its security properties; evaluate our prototype implementation; and conclude with a discussion of remaining challenges.

2 RELATED WORK

Encounter-based communication We first review prior work on encounter-based communication. Unlike enClosure, the prior work is limited to communication among devices that have directly encountered each other.

SMILE [29] and SmokeScreen [15] form encounters by negotiating a key among all devices within radio range at a given time. MeetUp [31] supports pairwise secure encounters, but it is not privacy-preserving: devices are trackable because encounters are authenticated using certificates that link a public key to a user profile picture. SMILE, SmokeScreen, and MeetUp provide encounters only in the context of specific applications, while enClosure provides a general-purpose communication platform supporting encounter-based group communication.

SDDR [27] forms secure encounters among devices within range of Bluetooth. The encounter protocol we use in enClosure builds on SDDR, but uses Bluetooth only for device discovery and instead performs a Diffie-Hellman (DH) key exchange via the Cloud. This protocol has the same security properties as SDDR, but forms encounters more quickly and with lower energy consumption. SDDR does not support group communication on the encounter graph, which is the focus of enClosure.

EnCore [5] provides a platform for mobile social apps based on EbC. Users can designate sets of direct encounters as part of a social event, negotiate group keys, and communicate within an event. enClosure instead provides a general platform for context-aware group communication among users connected by chains of encounters subject to spatial, temporal, and causality constraints.

MobiClique [32] combines social networks and Bluetooth encounters to form ad hoc social networks. Previously co-located users can share content, be notified of nearby users with matching profiles, exchange friend invitations in an OSN, or post messages to interest groups. However, MobiClique encounters are linkable and not privacy-preserving; users advertise their presence and profile to nearby devices.

Proximity-based communication Delay-tolerant networking [20] uses periods of co-location/connectivity of mobile devices to opportunistically forward data. EbC and enClosure instead use encounters to form communication endpoints, which can be used subsequently for authentication, encryption, and naming of communication over separate, untrusted communication channels, potentially long after the physical encounters have ended.

AirDrop [1], Android Beam [2], ShareIt [3], or FilesGo [4] are designed for singular, ad hoc interactions among presently co-located devices. Unless users remember to exchange contact information, further communication becomes impossible once the devices are out of range. McNamara et al. [30] address opportunistic media transfer among co-located devices. To accommodate long transfer times, the system predicts periods of co-location (e.g., during a joint public transit ride) based on devices' encounter histories. However, devices are linkable across encounters and disclose shared files.

ViewMap [25] allows authorities to identify information relevant to an accident among videos recorded by car dashcams, while preserving users' privacy. Users exchange their videos' fingerprints with nearby vehicles using short-range radio as proof-of-presence, and anonymously upload trajectories and received fingerprints. The system then builds a spatial map that allows it to solicit relevant videos. This application could be built easily on top of enClosure.

Proximity-based social networking Services like Foursquare, Facebook Places, and Google Latitude require users to check in with their current location. The service matches locations and notifies users of nearby people and resources. These services require users to reveal their identity and whereabouts to a third party.

Proximity-based profile matching E-SmallTalker [35] and D-Card [12] provide profile matching via Bloom filters among co-located devices. These devices can be owned by strangers (E-SmallTalker) or friends (D-Card). However, users disclose personal information with nearby users; security and privacy is not a focus of the work. Serendipity [19] facilitates interactions among co-located users through a centralized server, using Bluetooth and a Cloud service that stores user profiles. Users reveal their whereabouts and their preferences to the service performing the matching. FindU [28] offers interest matching among co-located devices and uses secure multi-party computation (SMC) to ensure that only the profile of the best matching nearby user is revealed. Dong et al. [18] and Distl et al. [17] also address privacy-preserving profile matching. Privacy-preserving attribute matching can be built on top of enClosure, but remains as future work.

Internet-of-Things The security and privacy of IoT devices is an active area of research [34]. enClosure provides spontaneous, untrackable, and authenticable group communication among smart and IoT devices connected in the encounter graph.

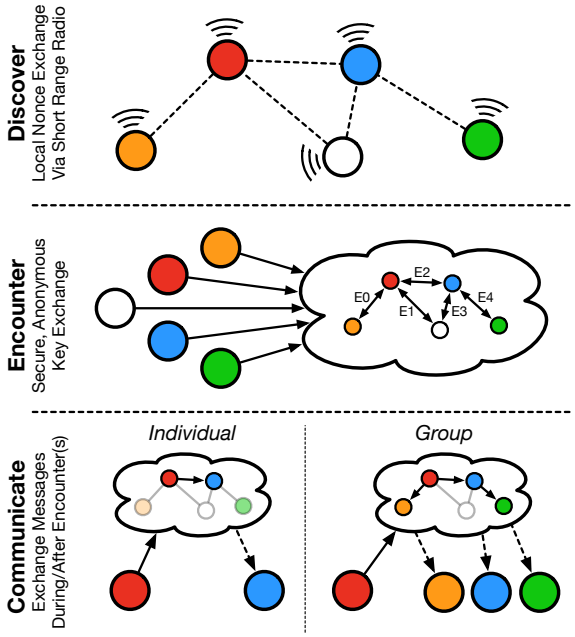

Figure 1: enClosure Overview: Users *discover* others, form *encounters*, and may *communicate* via encounter closures.

3 ENCLOSURE APPLICATIONS

Figure 1 presents an overview of enClosure. As in direct EbC, communication proceeds in phases. Devices initially discover others using short-range broadcast, and establish encounters associated with a specific location and time. During or after these encounters, devices connected in the encounter graph can communicate using an untrusted Cloud service, such as a third party key-value store. This simple, three-phase model has broad reach, and enables many new forms of contextual group communication that go far beyond direct EbC while retaining the same spontaneity, security, and privacy properties.

In the following, we sketch example scenarios and applications that can be supported by enClosure; in later sections, we discuss how exactly enClosure forwards these messages.

Communication among event attendees Attendees of a meeting or event may wish to trade tickets or swap seats, coordinate to purchase group tickets, offer and accept help, or identify individuals with a shared interest. Moreover, users may wish to communicate after the event has ended, e.g., to follow up with other attendees of a meeting, exchange photos or commentary about a shared journey, or contact a person they had missed at the event. enClosure enables such communication even among users who never encountered each other directly, as likely happens at large events.

Communication with nearby resources Users may wish to communicate with stationary resources in their vicinity. For instance, a user may receive up-to-date information about current conditions from an information beacon near a place she has visited, or retrieve information previously deposited at an edge storage device. Likewise, stationary devices can collect traffic statistics about passersby, possibly grouped by attributes that users are willing to share, and send messages containing relevant material or requests for participation in a survey. enClosure enables this communication even if a user was never close enough to a device to form a direct encounter.

Evidence of physical proximity A chain of encounters that occurred within a given region and period provides evidence of (past) physical proximity. For instance, event organizers can rely on this chain to distribute materials or coupons to attendees via encounters; similarly, receiving a message or friend request via a chain of encounters can serve as evidence that the sender and the receiver attended the same event. enClosure provides this evidence even for pairs of devices or users that never formed a direct encounter.

The above applications can be supported also by direct EbC, but only among devices that have had a direct encounter. As a result, the range of Bluetooth radio places an an arbitrary limit on the scope of possible communication. enClosure generalizes and extends support for these applications by extending the scope of communication to devices that never formed a direct encounter, e.g., because the diameter of the event or location exceeds the range of Bluetooth. Next, we describe new patterns of communication that enable applications not possible with direct EbC.

Virtual guest book Users can post an (anonymous) message to individuals who visit a given location at different times in the style of a virtual guest book. The messages can be seen by users who visit the site in the future and, if so desired, those who had visited in the past. Guest book locations could be tourist sites, restaurants, hotels, or even real estate on the market, and can be used to convey simple greetings, observations, or suggestions. Note that unless a sender and receiver visit the site at the same time, they cannot have a direct encounter. enClosure can deliver messages among devices connected by a path in the encounter graph via a stationary device at the location.

Health risk warning A health authority may want to warn individuals who could have contracted a disease, directly or indirectly, from a given sick patient. Consider the trajectory of the patient's device while she was contagious. The authority can send a message addressed to everyone reachable from any of the patient's encounters along the trajectory via a causal chain of encounters. Unlike a public broadcast, the authority can use this method to specifically target the set of individuals at risk, regardless of their subsequent travel patterns and location.

In a related scenario, given a specific area of contamination, the authority can warn anyone that might have visited that area during a certain period of time. Here, the message is forwarded via encounters that occurred within the specified area and time period. If the contagion can be passed on to others through proximity, then the message can also be addressed to all encounters that causally followed. The health risk application highlights the epidemic-style communication that enClosure naturally enables.

Lost and found Suppose Alice lost an item; she would like to send a message to individuals who might have found it. Consider Alice's trajectory since the last time or place she recalls having the item. Starting from the encounters her device had along this trajectory,

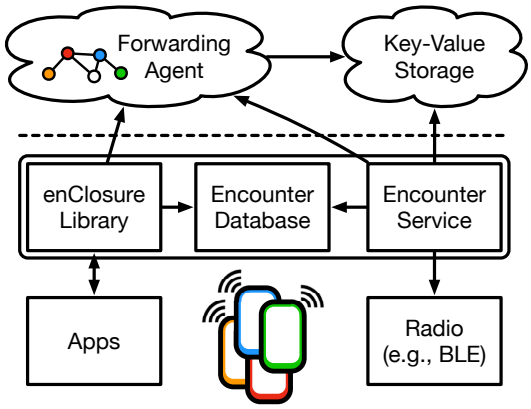

Figure 2: enClosure's high-level architecture, including components running locally on the users' devices (bottom) along with the Cloud services (top).

Alice can send a message to users who have been near any given point on her trajectory at any time during or after Alice was there.

Call for information/witnesses Authorities often wish to solicit information or witnesses related to an accident, crime, or missing person case. enClosure can be used to reach out to potential witnesses and to devices (e.g., security, dashboard, recreational cameras) that may have captured relevant information. Such calls can be targeted to persons/devices that were at or near the time and location of an accident, or near the trajectory of a person of interest.

For example, if a child Charlie goes missing, authorities may have a recent version of his device's encounter database (either because the device was found, or because the device regularly uploads its database into the Cloud, encrypted with escrowed keys). Law enforcement can send a message to individuals who may have seen Charlie or made other relevant observations. Consider Charlie's trajectory since his last known whereabouts. Starting from the encounters Charlie's device had along this trajectory, enClosure forwards the message to any device that has been in the vicinity of a given point on Charlie's trajectory[1] within some period around the time Charlie passed that point.

enClosure can support these and other applications, spontaneously and securely, without requiring users to reveal their identities, whereabouts, or communication, and without requiring them to consent to tracking their devices. Participating users merely have to run the enClosure service on their smart devices, which we show to be power-efficient in Section 8.

A common requirement of these enClosure-enabled applications is that communication partners are selected based on the spatial, temporal, or causal relationship of their encounters. This is a drastic departure from existing platforms where communication is formed around names, roles, or interests; this distinction is key to enClosure's strong privacy and security guarantees.

4 DESIGN PRELIMINARIES

We now describe enClosure's high-level architecture and the secure encounter formation protocol enClosure uses.

Figure 2 shows the high-level architecture of enClosure. enClosure client devices run the *encounter service* continuously in the background. The service forms secure encounters with nearby devices and adds them to the *encounter database*. The service relies on Bluetooth Low Energy (BLE) to discover nearby devices and relies on a Cloud key-value store to form and maintain encounters. Moreover, it periodically uploads the device's encounter database to the *forwarding agent* over a secure TLS channel.

Apps running on client devices who wish to use enClosure link to the *enClosure library*. The library provides a messaging API, which allows apps to poll for incoming messages and to send messages. In order to encrypt, authenticate, and address messages, the library consults the encounter database. It retrieves incoming messages from and places outgoing messages into the key-value store.

The key-value store holds encrypted messages in transit, as well as Diffie-Hellman parameters advertised by enClosure devices. Our prototype relies on the Microsoft Embedded Social (ES) platform as the key-value store.

The forwarding agent forwards messages according to the constraints in messages' headers and devices' encounter databases. As part of the forwarding, the agent retrieves, re-encrypts, and inserts messages from and to the key-value store. The agent executes in a trusted execution environment (Intel SGX in our prototype). Secure TLS channels are used for all communication between devices, forwarding agent, and key-value store.

4.1 Forming secure encounters

We briefly describe the protocol used by enClosure to form secure encounters among devices in BLE range. Devices continuously form encounters without user interaction; therefore, energy consumption is a focus of the protocol design.

The protocol is based on the BLE version [5] of the SDDR protocol [27]. Note that the particular choice of secure encounter formation protocol is not central to enClosure's contribution, namely group communication over encounter closures. Unlike the original SDDR implementation, our implementation uses BLE only to discover devices and exchange nonces. The Diffie-Hellman (DH) key exchange to compute the shared secret is performed via the key-value store for scalability and energy efficiency. In the original SDDR implementation, DH parameters are exchanged via BLE, requiring multiple advertisements because the parameters exceed the maximal size of a BLE advertisement. In contrast, enClosure advertises only a nonce, which fits within a single BLE advertisement. The exchange of DH parameters is done via the key-value store. Thus, our implementation can form an encounter even when two devices have received only a single advertisement from each other.

BLE supports periodic and energy-efficient broadcast of small amounts of data in the form of advertisements, which can be completely offloaded to the BLE controller. To discover nearby devices, the controller can be instructed to scan for advertisements from nearby devices. In addition to offloading advertisements, recent BLE controllers also support batched scanning, whereby advertisements

[1] Subject to connectivity in the encounter graph, as discussed in Sec. 5

from nearby devices are stored in a buffer that the CPU can process later on. Our implementation uses both mechanisms to reduce the frequency at which the CPU wakes up, as each wake-up consumes significant energy.

Device discovery and advertisement Time is divided into epochs. In epoch i, a device d maintains a private key $S_d^i = a$ and public key $P_d^i = g^a \mod p$. Device d's BLE controller advertises a nonce[2] $n_d^i = h(P_d^i)$, where $h(\cdot)$ is a secure hash function. At the same time, the device's BLE controller scans for advertisements from nearby devices using batch scanning while the device is sleeping.

Processing advertisements Devices periodically wake up to process advertisements received during the past batch period (e.g., the past 2 minutes). A device adds the advertisements to its *encounter database*, along with a timestamp, as *unconfirmed encounters*. Moreover, a mobile device records its current location in order to extend its *trajectory*, i.e., a sequence of the device's geographic coordinates. (A stationary device simply stores its coordinate once.)

Epoch change While processing every nth batch of advertisements, a device additionally performs an epoch change (e.g., $n = \lceil 15/2 \rceil$ for 15-minute epochs). The device d generates a new DH public/private key pair P_d^{i+1} and S_d^{i+1}, and instructs the BLE controller to advertise the new nonce and choose a new MAC address. Device d puts P_d^{i+1} into the key-value store with n_d^{i+1} as the key.

Confirming encounters Upon request by an application, or during an epoch change when there is a good Internet connection, a device performs a DH key exchange via the Cloud key-value store for some or all unconfirmed encounters. For every nonce n received over BLE, a device can look up the associated DH public key by querying for n in the key-value store. If and only if the hash of the retrieved DH public key matches the nonce received in the advertisement, the device completes the DH key exchange by computing the encounter secret E_s from the associated DH public key and its own DH private key at the time of the encounter. It produces the corresponding encounter ID $E_{id} = h(E_s)$.

Besides E_{id} and E_s, a device stores a timestamp corresponding to the beginning and the end of each encounter in its database. By joining an encounter's period with the device's trajectory, we can determine the trajectory covered by each encounter.

Linkability Devices change their advertised nonce and MAC address every epoch. Ignoring radio fingerprinting attacks [9, 23], which require specialized equipment, a device reveals no information to nearby radio listeners that could be used to re-identify the device across epochs. However, two devices that have a confirmed encounter can selectively recognize each others' advertisements, and thus avoid forming a new encounter in each epoch. Towards this end, during an epoch change, a device writes its new nonce into the key-value store under the ids of all existing encounters: for each existing encounter E that d has with a nearby device, d inserts $Enc(n_d^i)_{E_s}$ into the key-value store using E_{id} as the key, where $Enc(\cdot)_{E_s}$ encrypts its argument with shared secret E_s, and E_s and E_{id} are the shared secret and id associated with E. (In case a device has poor or intermittent connectivity, it can choose to defer the Cloud communication.) This allows encounter peers to recognize d's advertisements in the new epoch: when processing a received advertisement not seen before, a device first checks if the advertised nonce matches one in the key-value store under the E_{id} of an existing encounter; if so, it associates the nonce with the encounter in its database and does not form a new encounter.

[2]Not a nonce in the usual sense, as it is broadcast throughout an epoch and received by any listener during that epoch.

5 ENCLOSURE MESSAGING

We now describe how messages are forwarded in enClosure. We begin with direct messaging across a single encounter for illustrative purposes, even though it is not our focus. The sender names the message's destination using a location and time at which an encounter occurred. The library finds a matching encounter with id E_{id} in the database. It adds a message authentication code (MAC) keyed with E_s, encrypts the message with E_s, and inserts the encrypted message into the key-value store using E_{id} as the key.

Devices periodically poll the key-value store for incoming messages via any of their encounters. To avoid unwanted communication, users can install filters so that they see only messages received via encounters they had at particular times and places or with particular forwarding constraints.

Direct messaging, as described above, requires that sender and receiver share an encounter. enClosure messaging lifts this restriction and enables communication among devices that are connected indirectly via a path in the encounter graph. enClosure messaging takes advantage of the encounter graph structure to address messages to encounter closures subject to time, space, causality, and path length constraints. As we shall see later in this section, enClosure messages are always forwarded in two physical hops for each receiver via the forwarding agent; however, the communication partners may be connected via longer paths in the encounter graph. Next, we describe the messaging protocol and its properties.

Message header enClosure messages are forwarded according to a constraint provided by the sender in the message's header. The header contains the following information: the maximal path length h (where $h = 1$ for direct messages), the maximal fan-out f, the expiration time t_E, and the forwarding constraint.

h is the maximal distance in the encounter graph over which sender and receiver can be connected for the message to be delivered. f is the maximal fan-out in the spanning tree connecting senders and receivers in the encounter graph. The combination of h and f is meant to provide a backstop for the number of message deliveries that a single group message can generate; it is not meant to specify the set of receivers. After the expiration time t_E, a message is no longer forwarded or buffered. The forwarding constraint is described next. All encounters in a tree connecting senders and receivers must meet this constraint.

Forwarding constraint A message's forwarding constraint is a combination of one or more of the following types of constraints:

Space-time intersection Defined by a space-time region $R = (lat_1, lat_2, lon_1, lon_2, ele_1, ele_2, t_1, t_2)$ and a minimal intersection period τ. R is a hypercube defined by latitude range $[lat_1, lat_2]$, longitude range $[lon_1, lon_2]$, elevation range $[ele_1,$

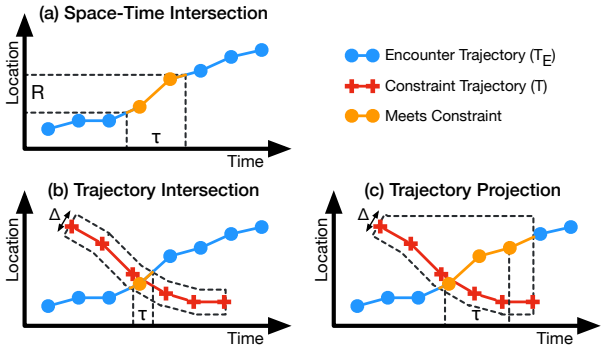

Figure 3: Example encounter meeting various enClosure forwarding constraints. For illustrative purposes, location is projected into one dimension.

ele_2], and period of time $[t_1, t_2]$. Let $T_E = L_{E,t_1}, ..., L_{E,t_n}$ be the trajectory of an encounter E, where L_{E,t_i} denotes the longitude, latitude, and elevation of encounter E at time t_i. If an encounter E's trajectory T_E intersects R for at least τ seconds, then the message is forwarded via E (Figure 3 a).

Space-time intersection can be used, for instance, to forward a message to every one who attended a stationary event or visited a location with a virtual guest book. If the event is large or there are many devices in a tight space, not all devices will necessarily form a direct encounter and they will therefore need forwarding across encounter closures in order to communicate.

Trajectory intersection Defined by a space-time trajectory T, a scalar distance Δ, and a minimal intersection period τ. An encounter E's trajectory T_E must be within Δ of T for at least τ seconds in order for the message to be forwarded via encounter E (Figure 3 b).

Trajectory intersection can be used, for instance, to send a message to all passengers on a particular vehicle (e.g., all those aboard a cruise ship), where T defines the vehicle's paths and Δ accounts for the diameter of the vehicle plus any slack required due to limited localization accuracy. The constraint can also be used to reach out to people who were sufficiently close to a missing person's trajectory that they may have relevant information.

Trajectory projection Defined by a space-time trajectory T and a scalar distance Δ. A message is forwarded via encounter E if E's trajectory is within Δ of a location on T's path at any time during or after T crossed that location. More precisely, if there exists a point L_{T,t_1} on T's trajectory and a point L_{E,t_2} on E's trajectory, such that the spatial distance between the two points is within Δ and $t_1 \leq t_2$, then the message is forwarded via E (Figure 3 c).

Trajectory projection can be used to send a message to devices that were near a location on a device d's trajectory during or after d passed that location. It can be used, for instance, to contact individuals who may have seen or found an item lost by another individual.

Causality Defined by the transitive closure of the "encountered-before" relation. The message is forwarded via encounter E_2 if it was received via an encounter E_1 that began before E_2 ended.

Causality can be used to forward a message via those encounters that may be causally dependent on the encounter via which the message was received; for instance, those who may have contracted a virus or learned a particular piece of information that may have arrived via an earlier encounter.

Powerful multicast semantics can be achieved by combining one of space-time intersection, trajectory intersection, or trajectory projection with the optional causality constraint. Table 1 summarizes the forwarding constraints and parameters used by each application discussed in Section 3. This shows that the diverse requirements of different, powerful communication patterns can be achieved with a combination of the forwarding constraints defined above and a suitable choice of values for a few parameters. Recall that h and f provide a backstop for message duplication and should over-approximate the width and depth of the tree connecting sender and intended receivers in the encounter graph. E.g., for a virtual guest book, we expect a deep but relatively narrow tree, while we expect a wide, shallow tree for a stationary event setting. In health-risk scenarios, we expect the tree to be both deep and wide; such a message would in any case require authorization by an authority.

Message forwarding Conceptually, enClosure messages could be forwarded physically hop-by-hop via receiver devices. However, forwarding messages via client devices has high energy and network costs, because every message copy has to be encrypted and sent by one client, then received and decrypted by another client device. It may also cause long delays when devices are unavailable for extended periods, and requires trusting receiver devices.

enClosure instead relies on a forwarding agent, which reduces energy consumption of client devices and achieves scalable message forwarding. The agent fetches one encrypted message copy from the sending device in one physical hop. It determines the receivers by including devices connected to the sender via encounters subject to the forwarding constraint. Finally, in a second physical hop, the agent encrypts and inserts a copy of the message individually into the key-value store for each receiver.

Upon receiving an enClosure message from a sender, the agent iteratively computes the set of receiver devices by consulting the encounter graph assembled from clients' individual encounter databases. The agent effectively performs a breadth-first search of the encounter graph, starting from the sending device, considering only encounters that meet the message's forwarding constraint, and respecting the message's fan-out and path length limits. The agent requires that the trajectories of an encounter as recorded by each encounter peer both match a message's forwarding constraint. This check yields robustness to incorrectly recorded encounter trajectories. It then encrypts a message copy for each receiver and inserts the copies into the key-value store.

After a message is forwarded, the agent stores a copy until the message's expiration time is reached. While a message is buffered, its forwarding constraints are checked against any new encounters and forwarded in the case of a match. Buffering is done on a best-effort basis: if space is lacking, the message body will be discarded.

Application	Parameters h	f	t_E	τ	Space-Time lat, lon, ele	$period$	Trajectory T	Δ	Type	Causal
Event:										
Stationary	↓	↑	**(1)**	**(2)**	Ev.Space	Ev.Period	-	-	-	No
Mobile	↓	↑	**(1)**	**(2)**	-	-	Ev.T	Ev.Δ	Int.	No
Guest book	↑	↓	**(3)**	0	Location	Timespan	-	-	-	No
Health risk*:										
Person	↑	↑	**(4)**	1s	-	-	**(5)**	0	Int.	Yes
Area	↑	↑	**(4)**	1s	Area	Risk period	-	-	-	**(6)**
Missing:										
Item	↓	↑	1d	0	-	-	**(7)**	0	Proj.	No
Person*	↑	↑	30d	1s	-	-	**(8)**	**(9)**	Int.	No

(1) Remaining event period
(2) Minimal attendance time
(3) Future portion of timespan
(4) Same as risk period
(5) Contagious person's trajectory
(6) Yes, if contagious; otherwise, no
(7) Trajectory since last had the item
(8) Trajectory since last known whereabouts
(9) Visual range
* Authorized by appropriate authority
↑ High ↓ Low

Table 1: enClosure applications: forwarding constraints and parameters

Message replies The sender of an enClosure group message may wish to enable replies from recipients, either privately to the sender or to the group. To facilitate replies, the sender includes in the message a key-value store key k and either a public crypto key for private replies or a shared group crypto key for group replies. Recipients encrypt their replies with the appropriate key and insert them into the key-value store using k, from where they can be fetched and decrypted by the sender or all recipients, as appropriate.

6 SECURITY PROPERTIES

We now define the threat model underlying enClosure's security properties, describe the properties, and examine the security limitations of our chosen threat model and system design.

6.1 Threat model

We assume that users and devices do not share the content of their encounter database with any party. Moreover, it is assumed users take appropriate measures to wipe the database along with other private data from a lost or stolen phone. Likewise, we assume that a device does not forward or otherwise share the BLE advertisements it receives with third parties, or advertise information over BLE on behalf of a third party. This assumption is reasonable for the proposed enClosure applications, which provide little or no incentive to collude to form additional encounters. Nevertheless, such applications may well exist, and we discuss ways to strengthen this threat model in Section 6.3.

The threat model also assumes that the operator of the forwarding agent is trusted to execute the agent often enough to ensure timely delivery of messages. Similarly, the operator of the key-value store is trusted to store encrypted messages and other data on a best-effort basis until they are deleted, and respond to requests often enough to ensure a timely delivery of messages and encounter formation. We assume that a *correct* client device records its encounter trajectories accurately, sends its encounter database to the forwarding agent, and sends messages via the forwarding agent.

The cryptographic primitives used are trusted, as is the execution environment in which the forwarding agent executes. Side-channel attacks against the forwarding agent, the enClosure client, or the encrypted communication are out of scope. The key-value store operator is trusted not to record client devices' IP addresses and mine this information for traffic patterns.

6.2 Messaging Properties

Subject to the threat model described above, enClosure has the following properties: The first two properties apply to direct messaging among devices that share a direct encounter, while the remaining properties hold for messaging among devices that do not share a direct encounter.

P0 (confidentiality): A sender can be sure that a message forwarded via encounter E can be decrypted only by the device that shares E. That device was encountered at the time and location recorded in the sender's database.

P1 (authenticity): A receiver can be sure that a message received via encounter E originates from the device that shares E. That device was encountered at the time and location in the receiver's database.

P2 (safety): A message sent by a correct device is received only by devices that have at least one encounter that satisfies the message's forwarding constraint.

Justification: Assume that the property does not hold. Let C be the chain of encounters along which the message was forwarded. We know that the sender is a correct device. It follows that somewhere along C, the message was forwarded to a device that does not meet the constraints. But neither a correct device nor the forwarding agent can do this within our threat model.

P3 (completeness): An enClosure message will eventually reach all correct devices that are connected to the sender by a chain of encounters that satisfy the message's forwarding constraint, if the message's expiration time is sufficiently late.

P4 (confidentiality): An enClosure message sent by a correct sender with forwarding constraint c can be decrypted only by devices that have at least one encounter that meets c.

P4 (authenticity): An enClosure message with forwarding constraint c received by a correct device originates from a device that has at least one encounter that satisfies c.

These strong properties hold despite the presence of devices that have recorded incorrect encounter trajectories. They result from the fact that (i) incorrect or malicious devices cannot forge encounters with correct devices; (ii) the forwarding agent does not forward messages via encounters that do not meet the constraints; and (iii) the agent forwards a message only if an encounter's trajectory as recorded by either peer both match the message's constraint. Thus enClosure can provide powerful group communications semantics

with strong properties, without requiring users and devices to exchange identities, consent to tracking, disclose their whereabouts or the contents of their communication.

It is important to note that enClosure identifies communication partners indirectly via closures of the encounter graph subject to times, locations, and causality, which gives it expressive power different from other forms of group communication. The notions of confidentiality and authenticity accordingly refer to partners connected by chains of encounters. The fact that communication partners in enClosure are not named by their identity is precisely what allows users to stay anonymous and untrackable unless they explicitly reveal themselves. If an application requires assurances that it is talking to a specific device or user, other forms of communication are appropriate.

6.3 Limitations and extensions

Next, we discuss limitations and possible extensions of enClosure.

Encounter graph density In general, the efficacy of enClosure messaging is subject to sufficient density in the graph of encounters. For instance, a multicast to all devices that reach a particular location within the next hour depends on the uninterrupted presence of devices within radio range of the location to ensure that the message can be forwarded. This condition is trivially met if there is a stationary device within radio range of the location, but otherwise depends on the movement of devices. A dense deployment of stationary enClosure devices (which would result, e.g., from support for the enClosure protocol by WiFi base stations) would largely remove this limitation.

Cryptographically secure forwarding agent The enClosure prototype forwarding agent relies on a Trusted Execution Environment (TEE) (Intel SGX) to ensure the agent's integrity and the confidentiality of users' encounter histories. While state-of-the-art, SGX has known issues regarding side channels and requires trust in the chip vendor's implementation and key management. Instead of relying on a TEE, it may be possible to design an oblivious agent that operates on encrypted encounter histories and messages. The agent could use proxy re-encryption [8, 11, 22, 24] to forward messages without access to message cleartext. A cryptographically secure agent remains the subject of future work.

Reducing trust in client devices In enClosure's current threat model, we assume clients do not share received nonces or advertise each others' nonces. We believe the design can be extended to make such collusion unproductive. We briefly sketch preliminary ideas; a full exploration and design remains as future work.

First, we can organize a device's encounter history as a hash chain and use its top-level hash at the start of an epoch as the nonce advertised during the epoch. The hash chain covers a device's long-term public key, its trajectory, the nonces it received, and the DH parameters it uses in each epoch.

Second, when devices upload their encounter histories to the forwarding agent, the agent (i) verifies the hash chain and rejects uploads that are not a linear extension of the existing history; and, (ii) checks each received advertisement for consistency with the history of the advertising peer and rejects encounters that are not consistent. These checks commit each device, identified by its long-term public key, to a linear history and ensures encounters are consistent with the histories of both peers.

Third, if a device's public key is tied to a physical trust anchor such as a Trusted Platform Module (TPM), then the history is tied to a physical device. A user can be allowed to use a small number of physical devices and periodically upgrade/replace devices, but must commit to these devices/upgrades in the history, where they can be checked for plausibility. If a physical trust anchor is not available, the agent can still check each (virtual) device's trajectory for physical plausibility (e.g., speed, acceleration) and consistency with other devices' trajectories (e.g., advertisements it should have received along its trajectory).

7 ENCLOSURE PROTOTYPE

Our enClosure prototype supports client devices running Android 8.0 or above, and requires no privileges beyond those of a regular app. The encounter formation service and messaging library consist of 11K LoC of Java, as well as 5.5K LoC of C++ for the modified SDDR implementation. The encounter database is stored in the existing platform SQLite database.

The forwarding agent runs inside an Intel SGX Enclave and consists of 2741 LoC of C++, not including the Intel SGX SDK and SGX-compatible libraries for OpenSSL and SQLite. Running the forwarding agent inside an SGX enclave helps to reduce trust in the operator of the forwarding service. It enables clients to verify that the agent is running the expected software via remote attestation, and prevents a curious operator from learning devices' encounter databases, the contents of messages, and other sensitive information like traffic patterns.

Our simple prototype forwarding agent is currently single-threaded and subject to the size limitation of available SGX implementations. Future implementations of SGX will allow for enclaves with much more memory, which can help greatly in combination with a multi-threaded implementation of the agent. In principle, the forwarding service can be scaled also to many devices by sharding the encounter database, and to higher message throughput by replicating the shards. The general techniques to achieve such scaling are well known; actually scaling our prototype forwarding agent remains as future work.

Forwarding agents and devices communicate with the Microsoft Embedded Social (ES) platform, which we use as a key-value store with notifications in our prototype, via secure connections. The prototype relies on two ES abstractions: topics and notifications. Topics are used as a key-value store for messages, DH parameters, and other information related to encounters. enClosure clients rely on ES's in-app notifications to efficiently poll for incoming messages and other information on a large number of topics (corresponding to encounters). ES is designed to scale to hundreds of millions of users and billions of topics, and provides a robust backend for our prototype.

8 EVALUATION

We begin with an evaluation of the encounter formation protocol. We quantify cost using micro-benchmarks that measure the energy consumption of running our enClosure prototype as it records

encounters and communicates with Embedded Social. Next, we briefly evaluate the throughput of our SGX-based forwarding agent. Finally, we explore the utility of enClosure by evaluating various application scenarios. Our results are derived from trace-driven simulations over a real-world dataset, specifically the SNAP Gowalla dataset [13], and described in Section 8.2.

8.1 Cost of running enClosure

There are two principal components to the (local) cost of running enClosure: the energy consumed by Bluetooth scanning to detect devices and exchange nonces, and forming encounters via ES. We address these in turn.

For these experiments, we use seven Sony Xperia XZ1 devices running Android 8.0 with a Qualcomm Snapdragon 820, 64-bit processor, a 2700mAh Battery, and 4GB RAM in all our experiments. We also use up to 100 Raspberry Pi (3 Model B) devices for our high-device-density experiments. We use the Monsoon Mobile Device Power Monitor (Ver 1.15) to sample power consumption at a rate of 5kHz using a soldered resistance tap at the battery terminal.

Bluetooth Overhead Devices must continuously scan for other devices using Bluetooth as a precursor to forming encounters. Recall that we use Bluetooth Low Energy (BLE) in our prototype. Suppose devices advertise over BLE every 100ms, and the CPU processes BLE input every two minutes. Just finding BLE peers in this scenario (including the base cost of the phone remaining suspended, but not the cost of forming encounters) consumes, on average, 27.46mW of power. This level of power consumption could be sustained over two weeks using the Sony Xperia XZ1's 10400mWh battery, thus confirming that BLE does not impose undue overhead.

As a stress test, we also measured the robustness of Bluetooth discovery and advertising when the number of devices (suddenly) increases, as may be the case when the user attends an event or walks into a populated venue. There are several potential concerns: Bluetooth may not perform well in crowded venues, and detecting and processing many devices may consume too much power.

We emulated this scenario using multiple BLE devices. Since we had only seven phones running the enClosure stack, we used 100 Raspberry Pi devices, each advertising every 100ms on all channels (simulating enClosure devices), along with the enClosure capable devices. We conducted the experiment in a 180 seat classroom (10m × 25m); pairwise distances among devices ranged from 0.6m to 17.5m. We positioned the phones running the full enClosure stack at the front of the lecture hall, approximately 3m from the nearest advertiser and 20m from the furthest. The lecture hall also contained multiple active WiFi base-stations that share the BLE spectrum.

The high-level takeaway from our experiment is the following: using our default parameters (two minute batch scanning → two weeks of battery life), every device is detected in every scan. This is an extremely encouraging result: it shows that BLE can easily scale to (at least) 100 devices with nominal energy consumption, and that our available hardware resources were not close to scaling limits.

Forming Encounters Once devices have been locally detected using BLE, enClosure can form encounters by conveying the nonces to the ES cloud service and performing the protocol described in Section 4.1. In this section, we quantify the overhead of this component of enClosure. The enClosure overhead can be decomposed into two parts: the fixed overhead of running the enClosure app, which periodically reports to ES, and the overhead of local BLE scanning. Running the enClosure app in the background, and reporting to ES once every epoch[3] (15 minutes in our experiments) increases the average power consumption as shown in Table 2. However, not all of this increase is directly attributable to enClosure: our inspection of the underlying power consumption shows that many Android services—in particular, permanent services such as anti-virus scanners that run by default—piggyback onto enClosure wakeups due to Android's timer coalescing [21]. Indeed, simply waking up the phone for a null wakeup (which relinquishes the CPU as soon as the app is scheduled) every 15 minutes, without running enClosure, consumes on average 34.58mW of power.

BLE? (Y/N)	WiFi? (Y/N)	Encounters (#)	Avg Power (mW)	Mean Battery Life (hr)
N	N	-	19.99 ± 0.03	520.26
Y	N	-	27.46 ± 0.28	378.73
Y	Y	-	32.40 ± 0.80	321.06
Y	Y	6 (real)	40.11 ± 0.41	259.29
Y	Y	6 (sim)	40.64 ± 0.89	255.90
Y	Y	10 (sim)	41.47 ± 1.08	250.78
Y	Y	16 (sim)	41.97 ± 0.73	247.82
Y	Y	32 (sim)	41.81 ± 1.14	248.74
Y	Y	64 (sim)	41.87 ± 0.32	248.37

Table 2: Power consumption and battery life of ES-based encounter formation while varying the number of encounters formed per 15-min epoch with a 2-min batch interval

Batch Interval (min)	Avg Power (mW)	Mean Battery Life (hr)
0.5	48.21 ± 0.44	215.70
1	42.94 ± 0.55	242.22
2	41.47 ± 1.08	250.78
8	40.68 ± 0.53	255.67
16	37.22 ± 1.73	279.42

Table 3: Power consumption and extrapolated battery life when forming 10 ES-based encounters per 15-min epoch varying the batch interval

Table 2 shows the cost of forming encounters while varying the number of devices encountered. The top of the table contains power consumption numbers for cases where we do not form encounters, with different radios turned on and off. Since we only had seven devices to run enClosure, we simulate performing key exchanges and linking with more than six other devices by creating and inserting multiple (simulated) BLE advertisements into the advertisement set received by the enClosure library. These advertisements, marked as "sim" in the result, are then processed as any other. We have repeated these experiments for different epoch intervals (7.5 to 30 minutes), without significant change in the power consumption.

The results show that increasing the number of encounters formed has little impact on the average power consumed; indeed, the dominant factor is the frequency of CPU wakeups, evaluated in Table 3. Table 3 shows an expected negative linear correlation between the batch interval (wakeup period) and the average power consumption.

[3]Recall that devices change their BLE identifiers every epoch.

Forwarding Agent Throughput We perform a throughput evaluation of the enClosure forwarding agent running within a SGX compartment on a server with 4 Intel(R) Core(TM) i5-6600 CPUs (3.30GHz) and 32GB of RAM.

Our experiment repeatedly chooses a user to send a message, which is processed and forwarded (sequentially) via the forwarding agent; the agent searches the encounter graph for encounters matching the message's forwarding constraint, encrypts the message with the appropriate E_s for each encounter, and posts the message to the appropriate E_{id} channels via network calls to Embedded Social. Our forwarding agent has a message send rate of 950 messages per second over a graph derived from encounters of 52 users over a 12-day period from the Haggle encounter dataset [16].

The goal of our SGX prototype was not ultimate performance, but to demonstrate the feasibility of secure forwarding using hardware enclaves. As such, we do not believe the reported forwarding rate to be a limit but merely a lower bound. Common techniques, such as sharding the encounter database and parallelizing across shards, would likely provide near linear speedup.

8.2 Utility of enClosure Applications

In this section, we seek to quantify the potential benefit of enClosure applications. There are no large-scale deployments of encounter systems from which we could gather encounter data. Instead, we simulate enClosure applications over a synthesized encounter graph derived from the SNAP Gowalla dataset [13]. This dataset records user locations for 6.4 million public check-ins for 107k users of the Gowalla social network, and was collected between Feb. 2009 and Oct. 2010. While the dataset contains data from users worldwide, there is notable user concentration in Western Europe, and in larger cities in the United States.

Synthesizing encounters The Gowalla dataset only contains (location, timestamp) pairs in the form of user check-ins. We synthesize encounters over this base dataset by adding a duration to each check-in, and then declaring an encounter between two users if two durations overlap, and the users are within 50 meters of each other. Each duration is sampled, uniformly at random, from an exponential distribution with a mean of one hour (the duration is truncated if the sampled value overlaps with the user's next recorded check-in). We chose this distribution because meetings, meals, and other social events tend to last about an hour, and since users in the trace checked in explicitly, it is likely they attended a deliberate event. This synthesis results in 1.69M encounters between 65.9K users.

Unfortunately, this dataset is rather sparse, and in practice, we expect even a modest enClosure deployment to generate many orders of magnitude more encounters. However, even over this sparse dataset, we show that messaging via encounter closures provides benefits; we expect such benefits to multiply over denser encounter datasets. In our simulated application scenarios, we often pick initial users or locations in and around Austin, Texas due to the (relatively) high number of encounters in the dataset in Austin.

Sending messages to event participants A simple and natural application for encounter systems is to send messages to attendees of a physical event. However, direct encounters cannot capture participants for large events, where the geographic distance is larger

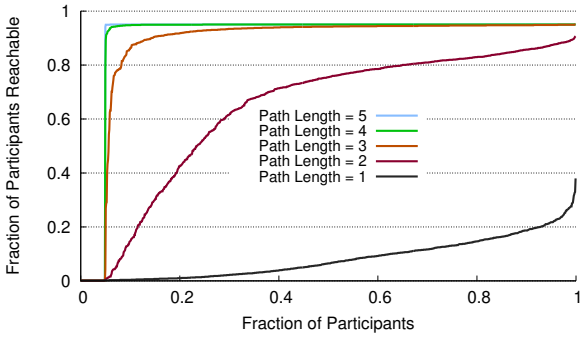

Figure 4: The CDF of reachable participants in the encounter graph by different users, while varying the maximal encounter path length from 1 to 5.

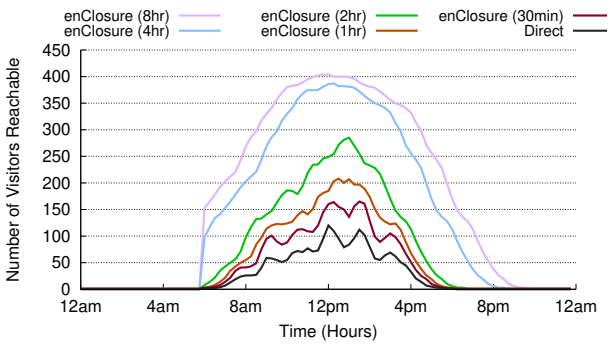

Figure 5: The number of visitors reachable by adding a message to the virtual guest book at a given time. In the 'Direct' case, the message will only reach the users currently in the area. For the 'enClosure' cases, messaging via encounter paths is used to allow others who visited the virtual guest book in the past (or will in the future) to see the message. The time window is denoted in parentheses.

than radio range. We simulate such a scenario using the Gowalla data from a 2km circle centered on the Austin Convention Center in downtown Austin, Texas from 6PM to 11PM on March 14, 2010. This event takes place during the 2010 South by Southwest (SXSW) music festival held at the Convention Center from March 12-21. The dataset contains 1108 total check-ins in the specified space-time region, which provides an upper-bound on the number of users who could be reached by enClosure. Figure 4 plots reachability of direct encounters compared to enClosure. Each curve is the CDF (over all users) of fraction of users that can be reached for increasing encounter path lengths. The 1-hop line corresponds to direct encounters, and shows that the vast majority of users (90%) would reach only 20% of all participants with direct encounters. Even the most connected users do not reach 40% of total participants directly. In contrast, even a single extra encounter hop allows more than half the users to reach over 70% of all participants. Encounter paths of length 3 allow more than 90% of users to reach close to 90% of all participants.

Virtual Guest Book Communication among devices that encounter the same location at different times fundamentally requires enClosure, since the devices cannot have a direct encounter. This

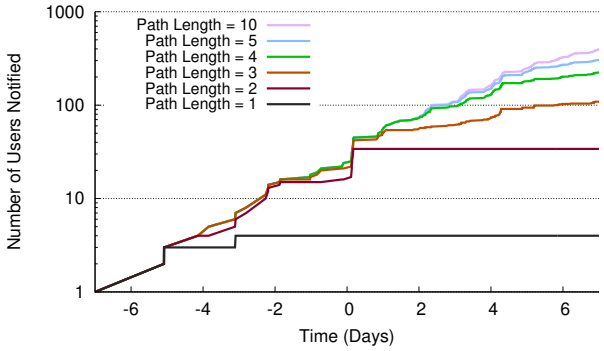

Figure 6: The number of users notified of the health-risk warning over time, while varying a constraint on the path length from the seed user.

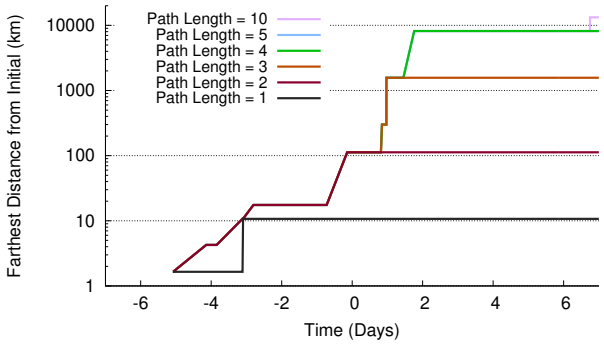

Figure 7: The farthest distance traveled by the health-risk warning over time, while varying a constraint on the path length from the seed user.

is true of an application such as a "virtual" guest book, where users at a certain location can sign the book and add a message that is viewable by future or past visitors. Moreover, connectivity in the encounter graph requires the continuous presence of devices at the location, which may be the case at busy venues but may not be true in general. A solution to this problem is to install a stationary device that forms encounters with all nearby users. Using enClosure, we can enable this "virtual" guest book with any commodity device running only the enClosure service.

In evaluating such a setup, we simulate the existence of the stationary device by the main entrance to the Austin Convention Center on March 14th (during the aforementioned SXSW festival). This device establishes encounters with all visitors that enter the convention center, and allow them to add entries to the guest book. As shown in Figure 5, enClosure's group message forwarding allows a visitor to leave a note at the Convention Center that will propagate to those who visited at different points in the day satisfying the space-time constraints of the guest book. For example, with direct messaging (1 hop), a user can only leave a note for other visitors present at the same time as themselves (a little over 100 people at peak hours); however, with enClosure's forwarding, the user's note will reach 300+ more visitors in an eight hour time window. A visitor could also leave messages for much longer periods.

Potential Infection Warning Perhaps the most powerful attribute of enClosure is to be able to address users over a large geographic area in a *targeted* manner based on their previous encounters. A novel and very useful application for such a primitive is for sending potential health-risk or infection warnings. Before we discuss this scenario in detail, it is important to clarify that we are not proposing that Bluetooth encounters necessarily map on to the spread of infectious diseases. However, current practice is to issue alerts over large areas (often large cities with millions of inhabitants) coupled with manual re-tracing of victims' whereabouts (e.g., [6, 14, 33]). enClosure can augment the efforts of healthcare professionals by providing a tool to *anonymously* identify potential patients and affected areas.

We simulate such a scenario using the Gowalla data by designating a single user in Austin, TX on March 14, 2010 as the "seed user". Any users that may have come in contact with this seed user should be notified: this corresponds to notifications sent to those with direct encounters with the seed. Additionally, since these users with direct encounters may themselves have been infected, it is important to spread the notification to all of their future encounters (and so on). Finally, we want to limit these notifications to a certain time frame from when the user could have been infected to when the infection would subside; in our case, we conservatively assume this time frame to be one week. In enClosure, we can express these notification parameters by constructing a single message with two constraints: 1) causal forwarding (i.e., sent to encounters that took place after receiving the message), and 2) a time constraint of one week since receiving the message.

We simulate the forwarding of such a notification message and plot the results in Figures 6 and 7. We treat the time of diagnosis as time 0 in these plots and focus in on the surrounding two-week window. In each of these plots, we vary the path length constraint. Figure 6 shows the number of users notified over time, starting initially with 1 (the seed user). Figure 7 shows the farthest distance the notification traveled over time. As shown, we can see that the seed user had only 4 (direct) encounters before their diagnosis at day 0, and never even left Austin, TX (having traveled only 10km at most). However, contagious diseases spread "virally", which is demonstrated by the growth in terms of number of users notified over time and increasing path lengths.

In this experiment, we picked the seed user specifically to demonstrate the "reach" of a single user even within our extremely sparse dataset (9399 users with encounters globally over the two week period in consideration). While this seed user was highly connected, they were not unique, and there were several other users with similar order of connectivity. Thus, even over our sparse dataset, this experiment is, we believe, sufficient to demonstrate the utility of the basic system. Indeed, blanket warnings to cover the "reach" of our seed user would have to include most of the contiguous United States, Western Europe, Japan, and other parts of the world. enClosure, however, can enable rapid and targeted communication with affected users and identification of affected areas.

9 CHALLENGES

We have described enClosure and its properties, applications, and basic techniques for group messaging across encounter closures as provided by enClosure. enClosure enables powerful group communication by leveraging temporal, spatial, and causal relationships between encounters. There are several interesting open questions, which we briefly discuss in this section.

Denial of Service enClosure messages are MACed and encrypted with the secret encounter key to protect their integrity and confidentiality. However, a communication channel, such as a Cloud key-value store, must ensure that an attacker is unable to delete, corrupt, or reorder other users' encrypted messages, or exhaust resources in the key-value store to deny service to others. Appropriate measures must also be in place to protect the key-value store from simple flooding DoS attacks. The enClosure prototype relies on a key-value store with a large key space, in which each key is *owned* by the device that created it, where the rate of key generations, puts, and gets is appropriately limited, and non-owners of a key can only perform non-destructive get operations on the key.

Unwanted communication Like any social communication platform, enClosure must deal with unwanted and inappropriate communication. In enClosure, this problem is more limited than in other communication systems like the general Internet since correspondents must be connected in the encounter graph. On the other hand, users may find unwanted communication particularly disturbing if they know it originates from an anonymous stranger who may be (or may have been) physically close.

A suitable mechanism for dealing with unwanted and inappropriate communication depends on the specific enClosure application built on top of enClosure. In general, appropriate mechanisms may include context-dependent whitelist and blacklist-based filters for unwanted communication; mechanisms to report and remove inappropriate or illegal content; as well as mechanisms to block repeat offenders. A technical challenge is to effectively block offending users without requiring all users to reveal a long-term identifier, which would enable the system to link users' actions.

Selecting communication partners enClosure is normally bundled with an app that uses enClosure for communication. Therefore, actions like selecting forwarding constraints are not directly exposed to users. Depending on the nature of an app, however, a user may have to select communication partners indirectly based on encounters. We see three broad ways in which this selection can be presented to users. First, in a spatial view, users browse their current location or past trajectory on a map, and choose a region on the map or a segment of their trajectory to select encounters. Second, in a temporal view, the users browse their calendar and choose an event or period to select encounters. Finally, in a contact view, the users choose one or more people from among their contacts to select the encounters that the users had in presence of these contacts. Combinations or subsets of these views are possible, depending on the app. For instance, a lost and found app might allow users to specify the time and/or place they last had the item.

Encounter database security enClosure stores a personal device's trajectory, which is sensitive, private information. Moreover, if an attacker were to gain access to the encounter databases of several users, she could determine the times and places where the users previously met by matching encounter ids across the databases. For this reason, the encounter database should be stored in encrypted form. To further reduce threats from malware, the database could be isolated from the rest of the system using a technology like ARM TrustZone [7]. Sensitive users may wish to limit their exposure in case of a breach by storing encounters only for a limited time, given that the utility of past encounters likely diminishes over time. Users concerned about being coerced to reveal their encounter database could additionally use a form of deniable encryption [10, 26] to conceal the true extent of their encounter database.

Encounter graph mining The encounter graph lends enClosure its powerful communication semantics and strong properties, and also contains additional rich information. Mining this graph presents an opportunity, for instance, to establish evidence of presence or independent identity, to study geographic traffic patterns and social dynamics, and for forensics. At the same time, allowing such access presents a risk to citizens' privacy. The graph as a whole, although materialized in the forwarding agent, is not directly accessible by any single party. Nevertheless, an attacker could try to mine the graph from different vantage points by sending enClosure messages with different forwarding constraints and observe where they are received, similar to the way traceroute can be used to explore the physical Internet. Understanding how to prevent such mining for illicit purposes and enabling controlled access for legitimate purposes is a fascinating subject for future work.

10 CONCLUSION

In this paper, we describe and evaluate enClosure, a novel type of group communication via encounter closures. enClosure allows messages to be addressed using spatial, temporal, and causality constraints that capture a wide variety of application scenarios.

We discuss how to build enClosure on top of existing secure, privacy-preserving encounter formation between co-located users. We evaluate the feasibility of continuous encounter formation, and show that enClosure can easily be supported using current mobile hardware. Finally, using trace-based simulations, we model real-world application scenarios that benefit from enClosure's messaging via encounter closures. Even though our simulations are based on a sparse dataset, our results show that enClosure is powerful, and we expect its utility to grow as encounter systems are deployed.

ACKNOWLEDGMENTS

We would like to thank our shepherd, Urs Hengartner, and the anonymous reviewers for their valuable feedback. The work was supported in part by the European Research Council (ERC Synergy imPACT 610150), the German Science Foundation (DFG CRC 1223), and the US National Science Foundation (NSF CNS 1526635 and NSF CNS 1314857).

REFERENCES

[1] Airdrop. https://support.apple.com/en-us/HT204144, 2011. Accessed: 2018-01-21.

[2] Android Beam. https://developer.android.com/guide/topics/connectivity/nfc/nfc.html#p2p, 2011. Accessed: 2018-01-21.

[3] ShareIt. http://www.ushareit.com/, 2014. Accessed: 2018-01-21.

[4] FilesGo. https://filesgo.google.com/, 2017. Accessed: 2018-01-21.

[5] P. Aditya, V. Erdélyi, M. Lentz, E. Shi, B. Bhattacharjee, and P. Druschel. Encore: Private, context-based communication for mobile social apps. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 135–148, New York, NY, USA, 2014. ACM.

[6] R. Allen and S. Cervenka. Travelers who landed in Detroit, Newark, Memphis pop up with measles. https://eu.freep.com/story/news/nation-now/2018/03/14/detroit-traveler-measles/425446002/. Accessed: 2019-04-15.

[7] ARM. ARM Trustzone. https://www.arm.com/products/security-on-arm/trustzone. Accessed: 2018-01-21.

[8] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, Feb. 2006.

[9] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless Device Identification with Radiometric Signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, 2008.

[10] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *Annual International Cryptology Conference*, pages 90–104. Springer, 1997.

[11] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 185–194, New York, NY, USA, 2007. ACM.

[12] A. C. Champion, B. Zhang, J. Teng, and Z. Yang. D-card: A distributed mobile phone based system for relaying verified friendships. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 696–701. IEEE, 2011.

[13] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1082–1090, New York, NY, USA, 2011. ACM.

[14] E. Commission. Commission launches EIC horizon prize for early warning for epidemics. https://ec.europa.eu/info/news/commission-launches-eic-horizon-prize-early-warning-epidemics-2018-apr-26_en. Accessed: 2019-04-15.

[15] L. P. Cox, A. Dalton, and V. Marupadi. Smokescreen: Flexible privacy controls for presence-sharing. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, MobiSys '07, pages 233–245, New York, NY, USA, 2007. ACM.

[16] C. Diot, M. Martin, and N. Erik. Haggle project, 2004.

[17] B. Distl and S. Neuhaus. Social power for privacy protected opportunistic networks. In *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–8, Jan 2015.

[18] W. Dong, V. Dave, L. Qiu, and Y. Zhang. Secure friend discovery in mobile social networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1647–1655. IEEE, 2011.

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

[20] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 27–34, 2003.

[21] Google. Alarmmanager | android developers. https://developer.android.com/reference/android\/app/AlarmManager.html. Accessed: 2019-04-15.

[22] M. Green and G. Ateniese. Identity-based proxy re-encryption. In *Applied Cryptography and Network Security*, pages 288–306. Springer, 2007.

[23] J. Hall, M. Barbeau, and E. Kranakis. Detection of Transient in Radio Frequency Fingerprinting Using Signal Phase. In *Proceedings of the IASTED International Conference on Wireless and Optical Communications (WOC)*, 2003.

[24] Y. Kawai and K. Takashima. Fully-anonymous functional proxy-re-encryption. *IACR Cryptology EPrint Archive*, 2013:318, 2013.

[25] M. Kim, J. Lim, H. Yu, K. Kim, Y. Kim, and S.-B. Lee. Viewmap: Sharing private in-vehicle dashcam videos. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 163–176, 2017.

[26] M. Klonowski, P. Kubiak, and M. Kutyłowski. Practical deniable encryption. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 599–609. Springer, 2008.

[27] M. Lentz, V. Erdélyi, P. Aditya, E. Shi, B. Bhattacharjee, and P. Druschel. SDDR: Light-weight, secure mobile encounters. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, pages 925–940, Berkeley, CA, USA, 2014. USENIX Association.

[28] M. Li, S. Yu, N. Cao, and W. Lou. Privacy-preserving distributed profile matching in proximity-based mobile social networks. *IEEE Transactions on Wireless Communications*, 12(5):2024–2033, May 2013.

[29] J. Manweiler, R. Scudellari, and L. P. Cox. Smile: Encounter-based trust for mobile social services. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 246–255, New York, NY, USA, 2009. ACM.

[30] L. McNamara, C. Mascolo, and L. Capra. Media sharing based on colocation prediction in urban transport. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 58–69. ACM, 2008.

[31] A. Mohaien, D. F. Kune, E. Y. Vasserman, M. Kim, and Y. Kim. Secure encounter-based mobile social networks: Requirements, designs, and tradeoffs. *IEEE Transactions on Dependable and Secure Computing*, 10(6):380–393, Nov 2013.

[32] A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: Middleware for mobile social networking. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 49–54, New York, NY, USA, 2009. ACM.

[33] K. Redmond. Measles: New york city officials issue health warning after australian tourist leaves highly infectious virus trail. https://eu.usatoday.com/story/news/health/2018/02/27/measles-officials-issue-health-warning-after-australian-tourist-leaves-highly-infectious-virus-trail/376336002/. Accessed: 2019-04-15.

[34] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao. A survey on security and privacy issues in internet-of-things. *IEEE Internet of Things Journal*, 4(5), Oct 2017.

[35] Z. Yang, B. Zhang, J. Dai, A. C. Champion, D. Xuan, and D. Li. E-smalltalker: A distributed mobile system for social networking in physical proximity. In *2010 IEEE 30th International Conference on Distributed Computing Systems*, pages 468–477, June 2010.