EnCore: Private, Context-based Communication for Mobile Social Apps

Paarijaat Aditya[†] Viktor Erdélyi[†] Matthew Lentz[‡]
Elaine Shi[‡] Bobby Bhattacharjee[‡] Peter Druschel[†]
[†]Max Planck Institute for Software Systems [‡]University of Maryland

ABSTRACT

Mobile social apps provide sharing and networking opportunities based on a user's location, activity, and set of nearby users. A platform for these apps must meet a wide range of communication needs while ensuring users' control over their privacy. In this paper, we introduce EnCore, a mobile platform that builds on *secure encounters* between pairs of devices as a foundation for privacy-preserving communication. An encounter occurs whenever two devices are within Bluetooth radio range of each other, and generates a unique encounter ID and associated shared key. EnCore detects nearby users and resources, bootstraps named communication abstractions called *events* for groups of proximal users, and enables communication and sharing among event participants, while relying on existing network, storage and online social network services. At the same time, EnCore puts users in control of their privacy and the confidentiality of the information they share. Using an Android implementation of EnCore and an app for event-based communication and sharing, we evaluate EnCore's utility using a live testbed deployment with 35 users.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: Security and protection; D.4.4 [**Operating Systems**]: Communications Management—*Network communication*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Location-based services, Privacy, Mobile computing, Pervasive computing, Social networking, Proximity-based services

1. INTRODUCTION

Mobile social apps consider users' location, activity, and nearby devices to provide context-aware services; e.g., detecting the presence of friends (Highlight, Foursquare [5, 9]), sharing recommendations about sights, goods and services (Foursquare), sharing content or gossip (FireChat, Whisper, Secret [4, 16, 19]), gaming (Nintendo 3DS, Sony PlayStation Vita [15, 17]), and connecting strangers who met but failed to exchange contact details (SMILE, Smokescreen [29, 49]). Users of these increasingly popular apps are exposed to various privacy risks. Most currently deployed mobile social apps rely on a trusted cloud service [5, 9] to match and relay information, requiring users to reveal their whereabouts, the perils of which have been extensively noted [7, 24, 27, 52, 59].

Some recent apps [1, 4] additionally use device-to-device (D2D) communication via short-range radio (e.g., Bluetooth, Wi-Fi Direct). D2D communication permits new capabilities: first, devices can precisely identify nearby devices, enabling powerful ad hoc communication and sharing. Second, D2D enables devices to create pairwise shared keys, which can be used to bootstrap secure and private communication without a trusted broker.

Recognizing this opportunity, new secure D2D handshake protocols, such as SMILE [49], SmokeScreen [29] and SDDR [47] have been developed. Our own prior work, SDDR, provides a *secure encounter* abstraction: pairs of co-located devices establish a unique encounter ID and associated shared key using D2D communication, which encounter peers can subsequently use for secure communication. While specific apps have been built using encounters [29, 49], no platform exists that relies on encounters to enable a wide range of privacy-preserving mobile social communication and sharing.

In this paper, we leverage the notion of addressable secure encounters introduced in SDDR to build EnCore, a communication platform that provides powerful new capabilities to mobile social apps. Using EnCore, apps can:

- Rely on encounters to conveniently and securely bootstrap *events*, which represent socially meaningful groups of proximal users and are associated with inferred context and user annotations.
- Send, receive, share, organize and search information and contacts by referring to events by their name, time or location, while maintaining confidentiality and full control over participants' anonymity and linkability.
- Use *conduits* to distribute and store information within events, before, during and after the actual social event. Current conduits rely on e-mail, Dropbox and Facebook.

MobiSys'14, June 16–19, 2014, Bretton Woods, New Hampshire, USA.
ACM 978-1-4503-2793-0/14/06.
http://dx.doi.org/10.1145/2594368.2594374.

To illustrate the space of apps supported by EnCore, consider a scenario of tourists visiting a site. While there, visitors wish to share live recommendations on nearby sights, shows to attend and eateries to try, but do not wish to reveal any (long-term) linkable information about themselves. If, unbeknownst to them, a friend or person with a shared interest is in the area, they would like to be notified, yet they wish to remain anonymous to all others. At a later time, attendees may like to share content (e.g., photos) and commentary related to the visit, but only with those who were there. Lastly, some might wish to follow up with a special person they met but failed to exchange contact information with. EnCore supports all these capabilities and more.

The primary contributions of this paper are as follows:

- We present the design of EnCore and its implementation on Android devices.
- We demonstrate EnCore's capabilities through *Context*, an Android application that provides communication, sharing, collaboration and organization based on events. The application was shaped by user feedback from a series of testbed deployments.
- We implement a variant of the SDDR protocol over Bluetooth 4.0. SDDR-4 takes full advantage of the new broadcast and low-energy features of Bluetooth 4.0, and is compatible with existing Bluetooth 2.1 accessories without compromising privacy.
- We report on a series of live deployments of *Context* and EnCore, with 35 users at MPI-SWS.

The live deployments described in this paper cover only a subset of the scenarios supported by EnCore, namely those involving colleagues within an organization. Events involving complete strangers, on the other hand, would require physical gatherings of people who do not know each other and who run an app based on EnCore. This requires wide adoption within a local community of sufficient size, or minimally a large event in which participants are incenticived to run the app. We hope to be able to arrange such deployments as part of future work.

Roadmap.

The rest of this paper is organized as follows: We sketch related work in Section 2, discuss EnCore's capabilities and requirements in Section 3, and present the design of EnCore in Section 4. The *Context* app, which makes the capabilities of EnCore available to mobile users, is described in Section 5. The Android implementation of our prototype is sketched in Section 6. We present the results of our experiments and testbed deployment in Section 7, discuss risks and challenges in Section 8 and conclude in Section 9.

2. RELATED WORK

Mobile social apps Most currently deployed mobile social apps like Foursquare [5], Whisper [19] and Highlight [9] rely on a cloud service to match co-located devices and relay data among them. Users must trust the provider with their whereabouts, activities and social encounters.

More recent systems like LoKast [11], AllJoyn [1], Haggle [8, 57] and Musubi [32], as well as lost-and-found apps like Tile [18], use D2D radio communication, which enables infrastructure-independent and accurate detection of nearby devices (e.g., those within Bluetooth range). In principle, these systems could be designed so that users do not have to trust the cloud provider with their sensitive data. Unfortunately, once Bluetooth discoverability is enabled, devices can be tracked even when they are not actively communicating, introducing a new threat to privacy. Unlike the tracking of cellular phones by mobile operators, such "Bluetooth surveillance" by stores and businesses is not regulated [42].

EnCore relies on D2D radio communication, but incorporates an efficient periodic MAC-address change protocol that ensures users cannot be tracked using their MAC address. The EnCore handshake protocol is provably secure and does not leak users' identity or profile information except to selected users.

The AirDrop [10] service in Apple's iOS 7 enables iPhone users to share content with nearby devices. AirDrop uses Bluetooth for device discovery and token setup, and an ad hoc Wi-Fi network to transfer data. AirDrop is designed for synchronous pairwise sharing among co-located users. Android Beam [2] is similar to AirDrop but relies on NFC [14] to initiate communication by physically placing devices back to back, and uses Bluetooth or Wi-Fi Direct [20] to transfer content. EnCore instead enables communication with all encountered EnCore devices, both during and after co-location. Moreover, EnCore prevents tracking, and supports anonymous and group communication.

Life-logging apps Friday [6] keeps an automated journal of user activities such as calls, SMSes, location history, photos taken and music history for browsing and sharing purposes. Memoto [13] is a life-logging camera that takes a picture every 30 seconds. The Funf framework used in the Social fMRI project [21] is a platform for social and behavioral sensing apps. Since all these services upload the collected data to the cloud, users have to trust the cloud provider with their private information.

Private mobile social communication systems SMILE [49] is a mobile "missed connections" application, which enables users to contact people they previously met, but for whom they do not have contact information. SMILE creates an identifier and an associated shared key for any set of devices that are within Bluetooth range at a given time. Users can subsequently exchange messages (encrypted with the shared key) anonymously through a cloud-based, untrusted mailbox associated with the encounter ID.

In SmokeScreen [29], devices periodically broadcast two types of messages, *clique signals* (CSs) and *opaque identifiers* (OIDs). CSs enable private presence sharing, announcing the device's presence to any nearby member of a mutually trusting clique of devices (e.g., friends). The sender's identity can be determined from the signal only by clique members, who share a secret. OIDs enable communication with strangers. A trusted broker can resolve OIDs to the identity of their sender, assuming that two or more devices agree to mutually reveal their identities. In comparison, EnCore supports anonymous communication with strangers without requiring a trusted broker.

SPATE [48] uses physical encounters among mobile devices to allow users to explicitly establish private communication channels, so that they can communicate and share data securely in the future. SPATE does not address anonymity, does not support communication among strangers who did not explicitly introduce their devices, and does not provide a way to address devices by referring to a shared context.

PIKE [22] is a key exchange protocol designed for secure proximity-based communication among the participants of an event. Keys are exchanged using an existing service like Google Calendar or Facebook, which require knowledge of the contact details for each participant. EnCore, on the other hand, leverages encounters to exchange keys with previously known and unknown participants, and without explicit user action.

SDDR: Secure Device Discovery and Recognition SDDR [47] builds on the encounter-based communication style introduced by SMILE, adding selective and unilaterally revocable linkability. The SDDR handshake protocol is provably secure, non-interactive and energy-efficient. SDDR attempts to form a pairwise encounter with each nearby device, establishing a shared key in the process. SDDR can also recognize specific users or users with specific attributes if both peers in an encounter agree to be recognized by each other, while remaining unlinkable by other devices. This *selective linkability* can be revoked and reinstated efficiently and unilaterally by each peer.

To prevent devices from being linked across encounters by their link-layer addresses, SDDR changes the MAC address every "epoch" (roughly every 15 minutes). However, periodic address changes are not natively supported in Bluetooth 2.1 and cause established connections to reset. As described in Section 6, our SDDR implementation over Bluetooth 4.0 maintains all of the security properties of SDDR over Bluetooth 2.1, and preserves interaction with legacy accessories and devices.

Privacy-preserving MAC protocols SlyFi [36] is a link layer protocol for 802.11 networks that obfuscates packet bits, including MAC address identifiers and management information, in order to prevent adversaries from identifying or tracking users in an application-independent manner. EnCore addresses the complementary concern of enabling anonymous, context-based communication based on encounters. EnCore, however, additionally includes a Bluetooth MAC address change protocol to prevent cross-encounter linking. Bluetooth 4.0 [3] is a new protocol that incorporates low-power, low-latency discovery and security extensions relevant to EnCore. We discuss Bluetooth 4.0 in Section 6.1.

Location privacy Several works investigate location privacy for mobile devices [35, 37, 38, 41, 56]. Roughly speaking, the following two classes of approaches have been proposed. The first class proposes to send fake or perturbed location data, or send location data at coarser granularity [35, 38, 41, 44, 56]. This class of approach essentially trades off utility with privacy. The second class of approaches does not require data obfuscation, but resorts to anonymity [37, 38, 40]. For example, Koi [37] sends unperturbed locations to a cloud server; however, the location is not linkable with a user's identity (assuming two non-colluding servers). In comparison with these approaches, EnCore achieves location privacy without relying on trusted, centralized infrastructure.

Device discovery Energy-efficient device discovery in wireless networks has been studied extensively [25,33,39,45, 60]. EnCore currently uses a simple, static device discovery scheme, but could easily incorporate the more sophisticated protocols in the literature. Other work aims to enable users to prove that they were in a particular location [46, 54]. EnCore addresses the orthogonal problem of allowing users to prove that they were in the vicinity of certain other devices. The Unmanaged Internet Architecture [34] (UIA) provides zero-configuration naming and routing for personal devices. While it shares with EnCore the goal of enabling seamless communication among personal devices, UIA is not concerned with the specific communication model and privacy needs of mobile social applications.

3. ENCORE: CAPABILITIES AND REQUIREMENTS

In this section, we describe EnCore's capabilities in light of the communication requirements of mobile social apps and the privacy needs of users. EnCore provides its capabilities without relying on a third-party provider that is entrusted with users' whereabouts, activities and social encounters.

3.1 Detecting nearby users and resources

A basic requirement of mobile social apps is the detection of nearby resources and users. EnCore's secure encounters enable this capability using D2D communication. Detecting nearby resources has several variants:

Discovering when a known friend is nearby Friends can be members of certain online social network circles (e.g., friends, family, colleagues), or specific users that have previously paired their devices. For privacy reasons, a user should be able to control discoverability by individuals or circles manually, and based on the present time, location, or activity. Moreover, a user's device should be unlinkable by all other devices.

Discovering relevant resources and nearby strangers that match a profile The profile might include interests (e.g., "tango") or relationship status (e.g., "single male age 27 seeking female"). For privacy reasons, a user should be able to control discoverability by individual profile attributes manually, and based on the present time, location, or activity. Moreover, an attribute should be visible only to devices that advertise a matching attribute.

Keeping a record of (strangers') devices encountered This record is useful to communicate and share information related to a shared experience, taking place in the present (e.g., sharing recommendations for menu items while at a restaurant) or in the past (e.g., sharing selected photos from a joint tour bus ride). For privacy reasons, this record must not contain personally identifying or linkable information.

3.2 Event-based communication/sharing

Mobile social apps enable communication among members of a social event like a meeting or gathering. A key abstraction in EnCore is an *event*, a set of encounters relevant to a social event along with inferred context and user annotations. Typically, an event includes a subset of a device's ongoing encounters at a given time, and a device may be part of multiple concurrent events. For instance, while at a restaurant, Alice's device may participate in a dinner event comprising encounters with each of the devices present at her dinner party. Concurrently, her device may be part of a restaurant event comprising encounters with other guests at the restaurant. Both events are socially meaningful, and may be used to share photos and notes about the dinner with her party, and menu suggestions with the other guests, respectively. Note that Alice's device may also encounter devices of people who pass by outside the restaurant, which are not part of any event.

EnCore is able to infer certain types of events automatically, and users can create named events manually by annotating specific encounters. Events occur naturally as users are presented with relevant encounter and context information. For instance, moviegoers at a theatre might wish to share movie recommendations on the spot, while participants of a sightseeing tour may wish to share selected photos and videos days later. Attendees at a conference might wish to virtually carry on a conversation started in the hallway, texting and sharing links long after the conference is over. Supporting events has the following requirements:

Ad hoc event creation The ability to set up an event without the inconvenience of having to pair mobile devices with every attendee or enumerating every attendee by their contact details. This capability lowers the bar for setting up communication and sharing related to an ad hoc event or meeting.

Event-based communication The ability to send, receive, share, organize and search information and contacts by referring to the time/location or name of the appropriate event. This capability makes it easy to communicate with people one has met on a particular occasion, without needing to remember everyone's name or contact details.

Furthermore, the platform must protect user's privacy and data confidentiality, leading to two additional requirements:

Privacy control To protect privacy, users must retain the option to participate in an event with full contact details, a permanent nickname ("Alice"), or under an unlinkable, one-time pseudonym. The former may be appropriate for a meeting with business partners at a conference, while the latter are appropriate for sharing content related to a shared activity with strangers.

Access control The ability to control access to the event is critical for private event-based communication. An event may be restricted to any subset of those physically present during a specific event, and may optionally include additional users who are invited by a member.

4. ENCORE DESIGN

In this section, we describe the services supported by the EnCore platform. Figure 1 depicts the various components of the EnCore architecture. EnCore uses the SDDR protocol to form D2D encounters, and store these in the EnCore database. The Event Generator component groups, under user direction, related encounters into socially relevant named events, and stores these in the database. Users use applications to communicate with event peers. Depending on the event specification and the type of content shared, the Routing module decides how to forward event invitations, content and notifications to the members of an event. The information is sent using Conduits, which rely on an existing communication, storage or OSN service to effect communication. Applications usually default to specific conduits for particular event and data types, e.g., Dropbox for video sharing. Before describing each of these components, we discuss EnCore's security properties and threat model.

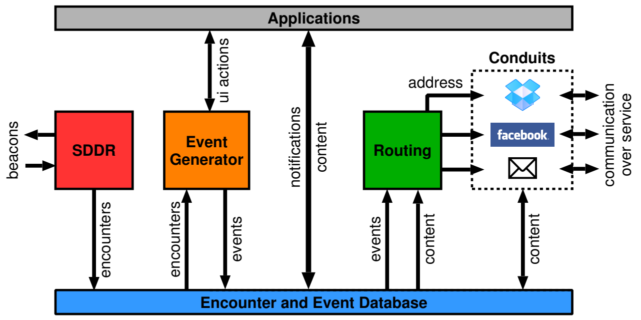

Figure 1: EnCore Architecture

4.1 EnCore security properties

4.1.1 Threat model

We assume that a subset of devices is controlled by attackers who participate in the EnCore protocol as peers and may also collude. Also, attackers can observe network communication and data stored in the Cloud. However, attackers cannot decrypt content without knowledge of the encryption key or invert cryptographic hashes. Furthermore, we assume that user devices are not compromised, i.e., attackers cannot learn honest users' private keys or the shared keys associated with encounters or events in which no attacker was a participant. Finally, honest users do not share event keys with non-members (users not part of an event).

We assume that user devices, including the operating system and any applications the user chooses to run, do not divulge information identifying or linking the device or user through EnCore conduits or other communication channels. (The EnCore protocols themselves do not leak identifiable information, down to the MAC layer.) Finally, we rule out radio fingerprinting attacks, which can identify a device by its unique RF signature [26]. Such attacks require sophisticated, non-standard signal capture hardware, and are outside our threat model.

4.1.2 Security properties

Under the assumptions stated above, EnCore provides the following security properties:

Bluetooth device unlinkability Attackers cannot track a legitimate user's device across Bluetooth radio contacts, unless the user's device remains in Bluetooth contact with some attacker's device for a continuous period that is never interrupted by more than two SDDR epoch changes.[1]

Encounter unlinkability/selective linkability Attackers cannot link different encounters with a user's

[1]Device unlinkability for Wi-Fi can be achieved using existing work like SlyFi [36]

device, unless the user has explicitly linked their device with an attacker's device and has not revoked the link.

Communication unlinkability Attackers cannot link communication or posts by a legitimate user in different events, unless the user has explicitly included identifying information, such as a nickname, in the posts.

Anonymity Attackers cannot learn the identity of a legitimate user or user's device with whom they share an event, unless the user explicitly reveals this identity.

Confidentiality Attackers cannot learn the communication content of events in which no attacker participates.

Authenticity Users can verify that the communication or content received in an event originates from a member.

We highlight that EnCore's threat model and security properties are mostly inherited from SDDR [47]. In principle, one could use a different platform, such as SMILE [49] or SmokeScreen [29] to support EnCore's functionality. Minimally, EnCore expects the underlying platform to discover nearby devices, form encounters and provide pairwise shared keys with them. SDDR additionally provides selective linkability and revocation, as well as Bluetooth unlinkability.

4.2 Encounters

EnCore uses a modified version of the SDDR [47] protocol for device discovery and for forming D2D encounters. Below we give a brief overview of how SDDR forms encounters and provides selective linkability. Further details, including SDDR's security guarantees and scalability, are available in [47].

Each device periodically performs a discovery (also known as an inquiry) to identify all nearby devices, collecting their MAC addresses and beacon messages in the process. Every device is also always discoverable, responding to incoming inquiry messages with information on how the discoverer can establish a connection (e.g., MAC address) and an additional payload, referred to as the beacon. This response is sent by the Bluetooth controller autonomously without requiring the attention of the main processor. Therefore, devices must only wake up to perform an inquiry. Otherwise, while simply discoverable, only the Bluetooth controller must be active, allowing the rest of the system to remain in a suspended state (consuming almost no energy).

Once SDDR receives the beacon(s), it forms an encounter with the remote device and computes a shared key. The beacon contains a Diffie-Hellman (DH) [31] public key which is used to compute this shared key.

While processing the beacon, SDDR additionally checks if the device belongs to a known, selectively linkable user. To support this, the beacon also includes a Bloom filter, which represents a set of salted hashes of secrets shared with the devices of linkable users. Two linkable devices advertise the same set member, which guarantees a match in the Bloom filters. The Bloom filters are padded to achieve a uniform load. Moreover, the salt is changed in every successive inquiry, which makes the probability of false positives quickly approach zero with each additional round in which the Bloom filters match. To a third (unlinked) device, on the other hand, the Bloom filters look like randomly changing sets of bits.

SDDR divides time into epochs (typically fifteen minutes long), during which the MAC address and DH public/private key pair remain constant. This allows the devices to track each other during an epoch, but remain unlinkable *across* epochs.

The original SDDR implementation used Bluetooth 2.1 to provide an efficient discovery and encounter formation implementation. Specifically, it encoded the beacons in the additional 240 bytes that the Bluetooth 2.1 Extended Inquiry Response (EIR) feature allows a device to include as part of the inquiry response. However, Bluetooth 2.1 does not support changing MAC addresses, which is *required* by SDDR; otherwise, users could simply be tracked by their MAC addresses regardless of the privacy provided by SDDR. As a result, the original SDDR implementation reset the Bluetooth controller with a new MAC address every epoch, every fifteen minutes or so. While this provided the necessary security guarantees, it also rendered the device unable to maintain long-term connections with other paired accessories such as headsets.

For use in EnCore, we migrated SDDR to Bluetooth 4.0, which provides native support for randomized, ephemeral MAC addresses. This feature enables EnCore to maintain compatibility with legacy accessories. However, the communication model supported by Bluetooth 4.0 is different from Bluetooth 2.1, necessitating an entirely new wire protocol and FEC-based message encoding scheme. Our design and implementation of SDDR over Bluetooth 4.0 is described in Section 6.

4.3 Events

Events are socially meaningful sets of encounters. The *Event Generator* creates events by selecting encounters that are taking (or took) place concurrently and form a social event meaningful to users. There are two methods for generating events: relying on explicit user input from the *Context* application, or using existing user annotations (e.g., the user's calendar entries). Once an event is created, the generator, using suitable conduits (Section 4.4), sends an invitation to all participating encounter peers, containing an event ID and a shared event key, which can be used for communication among event members. For privacy reasons, users are required to explicitly invite others for events inferred from their private calendar entries.

EnCore provides several methods by which users can create events: For small meetings, all participants tend to interact in close proximity with one another, and thus all devices form encounters with each other. Users can manually select appropriate encounters, using cues such as whether an encounter corresponds to a known user, or a received signal strength indicator (RSSI), which helps to distinguish between nearby and distant users.

For larger events and future events, it is inconvenient or impossible for one user to select all participants from the set of encounters they observe at the time of event creation. If the event is managed, and has a list of attendees, it is possible to bootstrap the EnCore event similar to PIKE [22], using Facebook or another existing registration system. However, unlike PIKE, EnCore can also handle ad hoc events. For these events, the event creator can specify a time period and geographic area, such that any devices within the specified space-time region is automatically invited to the event. This can be implemented by having each event member forward invitations over their encounters that meet the spatial and temporal constraints.

Evaluating such policies in a large scale deployment is part of ongoing work.

In designing EnCore, we chose not to use protocol means to disambiguate multiple EnCore events that correspond to the same social event. Thus, users are free to create multiple EnCore events for the same social event, or (somewhat more commonly for large events), a few users may end up creating their own EnCore event corresponding to the same social event. Our experience is that event peers themselves resolve this ambiguity by gravitating to one event, abandoning the others without requiring an arbitration protocol. (This was observed in our deployment as well, see section 7.2). Of course, applications atop EnCore may choose to provide their own arbitration protocol.

4.4 Communication

All application-level communication in EnCore occurs among the members of an event. Two types of components within EnCore are responsible for communication, *Conduits* and the *Router*.

Conduits are adapters to existing communication, storage and online social network services, and are used to convey information between event participants. Conduits accept messages or content and, depending on the type of conduit, either a list of encounter IDs (the communication endpoints for pairwise message transport) or an event ID (the rendezvous point for group communication and sharing). They convert the encounter IDs or event ID into addresses or names used by the underlying communication, storage, or OSN service that the conduit relies on. To provide secure communication among the members of an event, conduits normally encrypt the communication using either the shared encounter key(s) established during the handshake protocols, or the shared event key distributed during the event creation.

The *Router* component decides, based on the event specification and the type of information shared, how information is forwarded among event members. Three types of information are routed: event invitations, content, and notifications. If the conduit used for the event and information type supports multicast or shared storage, then the router delivers the information in one step, using the event ID as an address and the event key to encrypt. If the conduit supports pairwise communication, then the router sends the information to each member with which the local device shares an encounter, using the encounter IDs as addresses and the associated shared keys to encrypt. If not all pairs of event members share an encounter, then the routers on each member device forward the information to all of their local encounter peers that meet the event membership specification and have not already received it.

4.5 Security guarantees

Building EnCore on SDDR guarantees the security properties related to unlinkability. Since SDDR requires periodic MAC address changes, devices are not linkable at the Bluetooth layer unless the tracking device is present whenever the SDDR device changes addresses. Similarly, since SDDR ensures that the advertisements do not carry identifying information (except to linked users), devices remain unlinkable. The confidentiality and authenticity guarantees are provided by ensuring that all communication is encrypted and protected by a message authentication

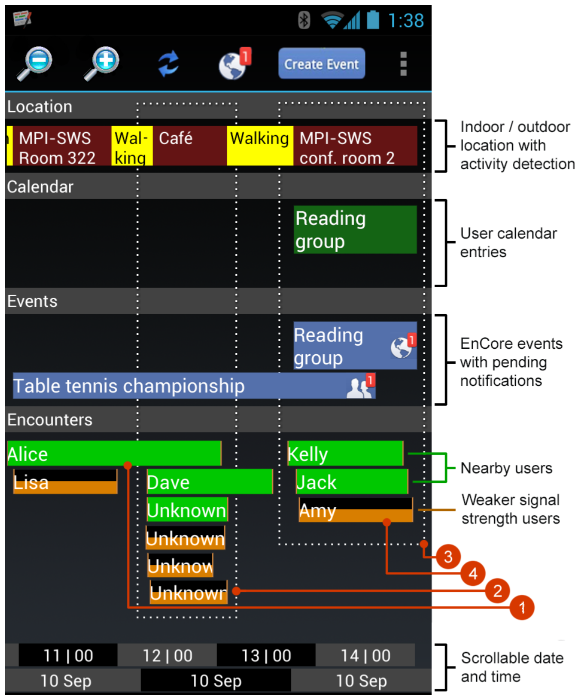

Figure 2: *Context* Timeline view (User names were changed for privacy)

code, using either an event key or an encounter key. Since only encounter or event peers possess the same shared keys, this ensures both confidentiality (event peers can decrypt) and authenticity (only encounter or event peers can post).

5. USING EVENTS WITH CONTEXT

We have developed an Android application called *Context* over EnCore. *Context* maintains a private record of the user's activities and social encounters, and allows users to communicate, share, collaborate, organize and search information and contacts using events. The design of *Context* was shaped significantly by user feedback during a series of testbed deployments within our institute community between September, 2012 and September, 2013

Even though *Context* still lacks the feature wealth, sleekness and visual polish of a commercial product, users in our institute-internal deployment have generally found *Context* useful, and have come up with creative uses for its capabilities. We provide quantitative details of *Context*'s usage in our deployment in Section 7.2, as well as qualitative user feedback in Section 8.1. In the rest of this section, we briefly describe the main functions provided by *Context* and their value to the user: browsing the user's timeline and identifying socially relevant encounters, managing events, posting and receiving information, and managing user linkability.

5.1 Browsing the timeline

Figure 2 shows a screenshot of *Context*'s timeline view for a hypothetical user Bob. Bob can navigate this view by scrolling, zooming or searching by keyword or date.

The timeline view shows encounters, events and calendar entries as horizontal bars spanning time intervals. For linked encounters, the peer's name is displayed. Anonymous encounters show "Unknown" as the peer name. The height and color of the encounter bar indicates signal strength and is a rough proxy for proximity. This view scrapes the user's calendar and displays previously scheduled entries. Any EnCore events are displayed in the events area. Events are marked with pending invitations or notifications (if any).

Selecting any UI element reveals more information about, and shows a menu of possible actions on, the element. For example, selecting an event highlights the participating encounters, allows the user to inspect or edit the event's metadata, invite more participants, or launch an application to browse the event's content. Selecting a location switches to a map-based view.

This simple linear view provides remarkable functionality: For instance, as shown by (1) on the Figure, by navigating back to this view, Bob can remind himself that he was with Alice at the table tennis championship before lunch on September 10, and eventually they walked to lunch together. The events pane shows that Bob was invited to the associated event and has a pending notification.

The rectangles (2) and (3) show how social events, both scheduled and impromptu, naturally line up vertically along the timeline. (2) shows that Dave joined Bob and Alice for lunch, and that there was someone else (unknown with high signal strength) nearby. This may be a sufficient hint for Bob to recall that Dave was with his guest at lunch. There were other lower signal strength encounters with unlinked users at lunch. Similarly, (3) shows a scheduled event, the Reading Group, that has a calendar entry and an associated EnCore event. Once again the vertical alignment of the high signal strength encounters with Kelly and Jack serve as reminder that they were at the reading group meeting. The encounter with Amy has low signal strength and likely is an artifact of her being in a nearby room but not at the reading group meeting.

5.2 Creating events

Users create events by touching the "Create Event" button and selecting a set of encounters to be included. If the event was previously scheduled in the calendar, its metadata (name, duration) is automatically imported; otherwise, the user can adjust the default duration inferred from the selected encounters, enter a name for the event, and optionally add the event to the calendar. Once the user confirms, the event is created and invitations are sent to the selected encounters. To support events where some of the users are not physically present (e.g. users attending an event virtually), event members can additionally invite any of their past encounters or known contacts to the event.

For more complex events or future events, the user can specify temporal and spatial (e.g., within the current building) constraints for included encounters. Future and transitive encounters that meet these constraints are invited automatically.

By default, the appropriate conduit to implement an event is chosen automatically. When all participants are linked encounters who provide Facebook account details (as is the case in our deployment at users' request), then a conduit is chosen that maps the event to a private Facebook event. Otherwise, a conduit is chosen that maps the event to a folder in Dropbox. The Dropbox conduit supports anonymous participants and provides the same basic sharing functionality, albeit without the integration and the sophisticated event presentation of Facebook.

These facilities make it easy to set up communication and sharing among a socially meaningful group of users in an ad hoc fashion. The event creator does not require contact details of the participants, and can include anonymous users via unlinked encounters.

5.3 Posting information

Context appears as a choice in Android's Share menu.[2] Therefore, any type of content can be selected (e.g., pictures and videos from the Android gallery, audio from a recording app, pin drops from a map app, text from a notes app) and shared via *Context*. Within *Context*, the user simply touches an event in which the content is to be posted.

As a convenient shortcut, users can post information directly from within *Context* and select encounters with whom the information should be shared, without creating an event. Internally, *Context* creates an event with default metadata to handle the posted information.

These facilities make it very convenient to send messages and share content with nearby or previously encountered users.

5.4 Receiving information

Notifications about incoming messages, posts or pending event invitations are shown as icons with red flags on the corresponding event, or on an encounter in the case of a message sent directly to an encounter. For instance, in Figure 2, there is a new post in the "Reading group" event, and a pending invitation to the "Table tennis championship" event. Notifications are also summarized in the notification center shown as an earth icon at the top of the screen. Touching it will scroll to the nearest event or encounter with a pending notification. Preference settings allow users to suppress notifications by type or source.

To respond to an event invitation, the user touches the event, optionally views the event's metadata, and then accepts, declines or defers the invitation. To read incoming messages or posts, the user selects the relevant event or encounter. Touching an event launches an external application (e.g., Facebook) to show the latest post in the event.

These facilities enable users to prioritize, filter, browse and navigate incoming information according to its context: event, encounter, time and location.

5.5 Controlling linkability

Context allows users to control the information revealed in an encounter in a variety of ways. The user can choose to reveal a linkable nickname or their real identity to selected peer devices. The linkable peers can be selected based on existing relationships in an online social network (e.g., Facebook) or a contact list, or by pairing devices individually. Moreover, linkability can be controlled based on the user's present location or time. For instance, users can choose to be linkable to colleagues only when in the office and not be linkable by anyone at certain times.

Recall that a Facebook private event is used by default for events among linkable encounters who provide Facebook

[2]Most Android content apps have a "Share" button, which opens a menu of applications through which the selected content can be shared.

details. User posts are linkable across such events. Users can use a separate Facebook account under a pseudonym for this purpose; in fact, all participants in our deployment use test accounts separate from their main Facebook account. To avoid linkability across events, the creator of an event can choose to use a Dropbox conduit instead, and users can decline invitations to Facebook-backed events if they so choose.

These facilities enable users to effectively control their privacy.

6. IMPLEMENTATION

In this section, we describe the implementation of EnCore and *Context*.

6.1 SDDR over Bluetooth 4.0

We have implemented a modified version of SDDR on the Android platform, using Bluetooth Low-Energy support instead of Bluetooth 2.1. We use Samsung Galaxy Nexus phones running Android 4.1.2 with the android-omap-tuna-3.0-jb-mr0 kernel. We modified this kernel by patching in Bluetooth 4.0 L2CAP communication support present in the Linux 3.7.7 kernel, as well as a power-related bug fix in the Bluetooth controller driver.

As described in [47], in order to provide selective linkability, SDDR devices store an advertised set and a listen set, each containing linkability values (or *link values*); if a value in device A's advertised set exists within device B's listen set, then B can detect A. Upon detection by the underlying radio, the SDDR protocol exchanges a beacon message containing a Diffie-Hellman (DH) [31] public key, and a set digest structure containing salted hashes of the advertised set values. Upon receiving a beacon from a remote device, an SDDR peer computes a shared DH key and checks to see if elements of its listen set are found in the newly advertised set.

The EnCore implementation of SDDR utilizes the broadcast feature of Bluetooth 4.0. Bluetooth 4.0 introduces new roles for devices, two of which are *advertisers* and *scanners*. Advertisers periodically broadcast small messages, containing a 31-byte payload, to nearby scanners who listen for messages. Unlike the exclusively pairwise communication in Bluetooth 2.1, advertisers in Bluetooth 4.0 can broadcast their messages to all nearby devices [3].

SDDR exchanges a beacon, consisting of a 192-bit DH public key and a Bloom filter containing the link values present in the advertised set; however, this beacon cannot fit into the 31-byte limit of Bluetooth 4.0 advertisement messages. Instead, we model the advertisement channel as a packet-erasure channel, i.e., packets are either received correctly or not at all, and send the beacon message, in segments, encoded using Reed-Solomon (RS) coding [53]. RS codes are optimal erasure-correction codes, meaning the receiver can reconstruct the original K data symbols after receiving any K of N total symbols.

Figure 3 illustrates our method for generating advertisements from the beacon information. Each advertisement contains an index to identify the RS and Bloom filter segments. We divide both the public key and the Bloom filter into segments, with each advertisement containing a single RS and Bloom filter segment. In order to produce the RS-coded segments, the coding matrix (in the form of a Vandermonde identity matrix [51]) is

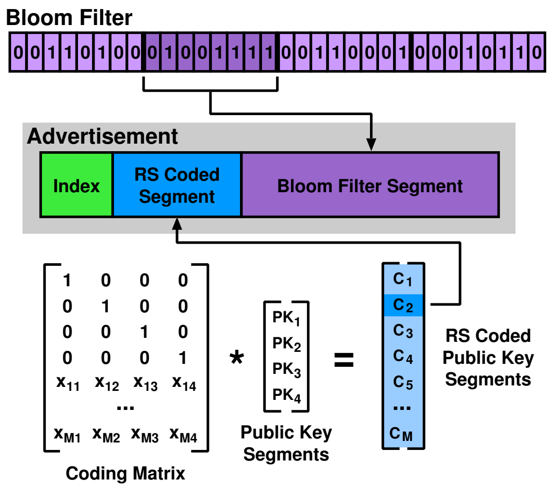

Figure 3: SDDR-4 beacon constructed from RS coded DH public key and Bloom filter segments

multiplied by the vector containing all segments of the public key. Note that the Bloom filter segments do not require forward error correction since if the receiver misses an advertisement, it can simply replace the segment with all 1's in the reconstructed Bloom filter. As more advertisements arrive, the effective false positive rate of the Bloom filter decreases.

The discovery protocol runs periodically (roughly every 15 seconds), acting as both an advertiser and scanner. Using the public key and Bloom filter generated at the start of each epoch, the device computes and starts to broadcast the next advertisement (according to the current index). Devices also scan to capture broadcasts from nearby devices, and report the set of discovered devices to the higher EnCore layers.

Once enough advertisements are received, the handshake protocol attempts to decode the remote device's public key and compute the shared key. Once decoded, the public key is used to query the Bloom filter (which may only be partially received) for all link values in the local listen set; as new Bloom filter fragments are received, the result converges to the true intersection as the number of false positives decreases.

6.2 Conduits and router

EnCore currently supports the following conduits:

SMTP conduit The SMTP conduit allows users to securely exchange e-mails with the participants of an event. The SMTP conduit allows any email client on the device to send a message to the email addresses associated with one or more encounter peers. If an encounter is linkable and has an associated email address, the message is simply sent to that address. If the encounter is unlinkable, then a one-time email address is derived from the encounter ID, and the message is sent to that address. The current implementation uses `mailinator.com` [12] for this purpose, which does not require user registrations and creates mailboxes on-the-fly as mail arrives for an address. The mail is public for all who can guess an email address, but this does not affect confidentiality since all EnCore mail is encrypted using an encounter or event key. The Mailinator conduit is limited by the the fact that Mailinator caches messages only for a few hours, and applications need to periodically resend messages to ensure persistence. Alternatively, one could easily setup a similar one time email system that does not delete messages.[3]

Dropbox conduit The Dropbox conduit converts an event ID into a folder name on Dropbox, and stores all content posted to the event into that folder, encrypted with the shared event key.

Facebook conduit This conduit associates an EnCore event with a private Facebook event. It requires that all participants in the event are linkable and provide details for a Facebook account. The event's participants appear in the Facebook event with the identity of the account they provided. Textual posts, comments, likes and photos are posted in the Facebook event in cleartext, to maintain the flexibility and convenience of the Facebook interface.[4] However, video and audio recording posted to the event are uploaded using the Dropbox conduit (encrypted with the shared event key), and a URL to that content is posted in the Facebook event.

Using Facebook allowed us to leverage the familiarity of users with its app. The Facebook conduit cannot support unlinkable users, which was irrelevant in our deployment. As part of ongoing work, we plan to recreate similar functionality within *Context* with the ability to create events amongst unlinkable users.

Router The current router implementation is limited to forwarding information within events in which all members share pairwise encounters. We are in the process of adding transitive forwarding. There has not been much demand for this feature so far, due to the relatively small size of our deployment and the types of events users have requested.

7. EVALUATION

We have evaluated EnCore using microbenchmarks of CPU usage and power consumption, and also via a series of user studies. All of our experiments used Samsung Galaxy Nexus devices, containing a 1.2 GHz dual-core ARM Cortex-A9 processor. We begin with a description of the microbenchmarks, and conclude with results from our latest user study.

7.1 Microbenchmarks

Protocol computation time We have measured the SDDR-4 handshake protocol computation time while varying the number of link values in the listen and advertised sets. We compare execution time to an implementation [30] of the JL10 scheme [43] for Private Set Intersection (PSI). Results are similar to the original SDDR protocol: For all set sizes, SDDR-4 executes over three orders of magnitude faster than standard PSI.

Energy consumption EnCore runs the SDDR protocol permanently in the background on a mobile device, and therefore power considerations have informed considerable parts of its design. We have collected extensive power traces of different parts of the EnCore protocol using the BattOr [55] power monitor. The BattOr samples the voltage and current across the battery terminals at a rate of 1 kHz, and makes this data available via (its own) USB connection. We collected this data for the SDDR-4 protocol variant, as described in Section 6.1. Detailed energy results for the original SDDR protocol can be found in [47].

Operation	Avg. Power (mW)	Energy (mWs)	When
Idle	1.73	-	-
Suspend/Wakeup	205.94	263.74	-
Bluetooth 2.1			
Inquiry Scan	140.46	1698.36	Every 30s
Name Request	144.97	96.26	-
SDDR-4			
Discovery	236.48	752.67	Every 13.5s
Advertisement	107.22	0.58	Every 700ms

Table 1: Power consumption for Bluetooth 2.1 and SDDR-4 operations. Entries without a value do not have well-defined durations.

Table 1 summarizes the energy consumed and the average power required for each of the different SDDR operations. The idle power consumption while the device is in airplane mode (no radios are enabled) is a negligible 1.73 mW. Since discoveries continually take place in our protocol, we include the power and energy consumption from suspending and waking up the device. As a point of comparison, we have included energy consumed by Bluetooth 2.1 inquiry scans and name requests.

Discoveries are the most expensive operations within SDDR, in part due to the overhead of suspending and waking up the device; for BT 4.0, this overhead makes up ~35% of the energy consumption. BT 4.0 advertisements consume very little energy ($<$ 1mWs), since only the Bluetooth controller (and not the main processor) must run; in addition, the broadcast takes place within a few milliseconds. Overall, the power consumption of SDDR-4 within EnCore is negligible, dominated by the overheads of standard Android services like activity detection and location determination (GPS). Power consumption increases by one order of magnitude once the screen is powered on and the user interacts with *Context*. Even with all features enabled continuously and users performing their normal activities, no user within our deployment had a device run out of battery mid-day during the testing period.

7.2 Live deployment

Our latest EnCore field deployment began on September 9, 2013, with 35 volunteers in the Saarbrücken office of MPI-SWS. The participants were staff members and researchers, and were informed about the purpose of the experiment and what data would be collected. Most of the participants (32 out of 35) were not directly involved in the project.

Deployment setup We provided each participant with a Galaxy Nexus phone running EnCore with *Context* and the Facebook app. All phones were configured with the account details of a different Facebook test account with a pseudonym. At users' request all phones were selectively linked with each other by default. Users were able to change the linkability settings, configure their personal calendars for display in *Context*, and change the pseudonym to their real name if they wished to do so (30 of the 35 users did). None

[3]We implemented such a system and used it during one of our initial deployments.

[4]Note that Facebook has access to cleartext posts; this can be avoided by using the Dropbox conduit or a private OSN platform like Persona [23] to share all information.

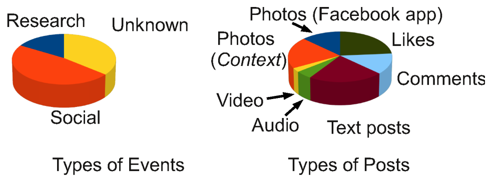

Figure 4: Types of events and posts

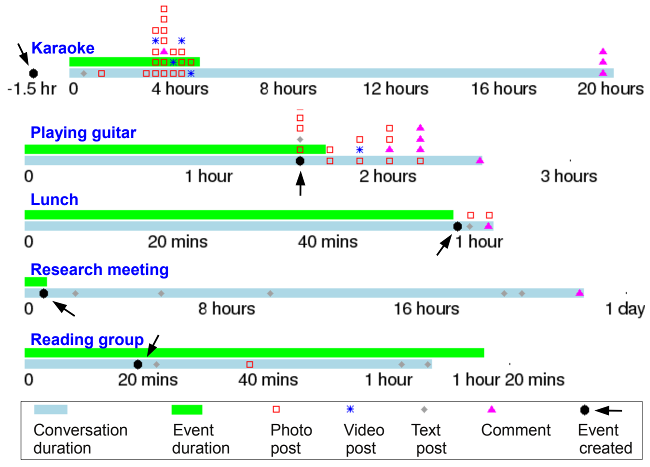

Figure 5: Timelines of five actual events. Each point shows the type of activity performed along with the time elapsed since the specified start time of the event

of the users modified the default linkability setting (i.e., link with all other users). This is not surprising since the deployment was carried out among mutually trusting users and linkability was limited to experimental devices only.

We requested users to carry the device, and encouraged them to use *Context* to create events, communicate and share content as they saw fit. On September 16, we installed an audio recording and a note taking application on each of the devices because several users requested it, in addition to the default camera, gallery, calendar and map applications already available on each device.

The phones ran EnCore, using the SDDR-4 protocol. The phones executed discoveries every 13.5 seconds and changed MAC addresses every 15 minutes.

Statistics from the deployment After the the first two weeks of the deployment, we collected statistics for all the events created, which we present below. The users' activity level was roughly bimodal. 17 users created fewer than three events and made fewer than five posts, while 18 users exceeded these numbers. Among the users not related to the project, the maximum number of events created by a single user was 12 (median 3) and the maximum number of posts created by a single user were 30 (median 4).

Event usage After removing from consideration a number of events that had been created by project members for demonstration purposes, a total of 128 events remain. We have divided these events into three categories based on their names: research meetings (16%), social events (48%), and unknown (36%). We classified an event as unknown if its purpose was not obvious from its name. Figure 4 shows

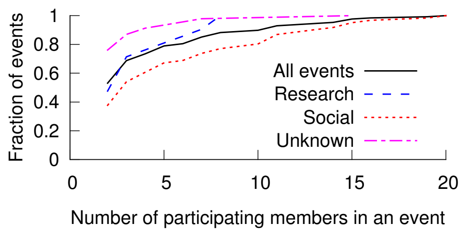

Figure 6: Distribution of event size

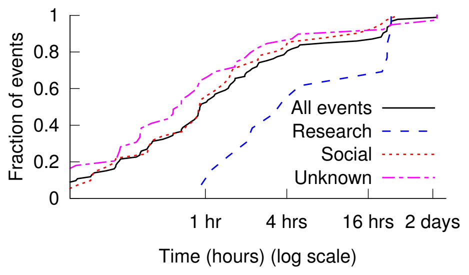

Figure 7: Distribution of conversation durations

the distribution of event types and Figure 5 presents the timelines of a selection of five actual events created during the deployment.

Based on informal feedback from users and our own observations, there was an interesting mix of expected and creative uses of events. Events were created for research gatherings, such as meetings, reading groups, etc., and used to exchange meeting notes, audio recordings, followup comments and links to related papers. The 'research meeting' and the 'reading group' events in Figure 5 show the activity timelines of two events in this category. Also, as seen in Figure 6, these events tend to contain a moderate number of participants, which is what we commonly observe for project/group meetings and reading groups in our institute.

Almost half of the events were created for social or informal gatherings, such as lunches, coffee breaks, sports activities, bus rides, karaoke events, etc., and used to share photos, videos and comments during and after the event. The 'karaoke', 'playing guitar' and 'lunch' events in Figure 5 are typical examples. With the 'karaoke' event we also observed an instance of users resolving multiple EnCore events created for the same social event by gravitating to one event (see Section 4.3). We observed that users stopped posting to one of the two events created for karaoke and continued their interactions on the other. Some creative uses include creating events to invite nearby people for coffee breaks, or to inform nearby people about leftover party food using a picture of the food. The number of members in social events ranges from 2 to 20 users (median 3), which is expected given the size of our deployment (Figure 6).

Figure 5 also highlights that events can be created after the associated event has ended, and that conversations tend to extend beyond the event duration. The former can be observed for the 'Lunch' event by comparing the time when the EnCore event was created (circular dot with an arrow) and the timespan of the actual social event (the first horizontal bar from the top for each event).

Figure 7 shows the distribution of conversation durations for different types of events. Even though most of the events did not have conversations longer than five hours, there were cases where users referred back to events they had attended in the past and posted content to them long after the actual event had finished.

Figure 4 shows the distribution of the types of posts within events. Note that the audio recording application was installed one week after the deployment had started; the proportion of audio posts might have been higher had it been installed from the beginning. A large portion of photos were uploaded directly from *Context*, rather than via the Facebook application, which suggests that users found it convenient to refer to an event directly from *Context*'s timeline.

Summary EnCore and *Context* provide a basis for exploring secure ad hoc interactions. Both the analysis of data from the deployment and personal user feedback show that real users find the paradigm useful and found new ways to collaborate and share with colleagues using *Context*.

8. DISCUSSION

In this section, we describe the qualitative feedback we received from our users, and discuss the remaining risks and challenges.

8.1 Qualitative user feedback

Quantitative performance evaluations are often inadequate in capturing the utility of new functionality. User engagement can be an important metric, and here we describe the qualitative feedback we received from our users, both during and after the test deployments. At the end our our latest deployment, many of our users expressed an interest in using the system on a permanent basis once we have a version they can install on their primary devices. We believe this is encouraging since it shows that users (albeit highly technically proficient ones) find the system useful. In the rest of this section, we discuss features that users have requested. We believe feature requests are illuminating. While they obviously point out shortcomings in the existing system, they also point to innovation enabled by, and creative use of, EnCore's capabilities, some of which were not anticipated by the design team.

Support for sharing audio recordings and import/export of events to/from calendar was requested by users and rolled out during the last deployment. Various requests were related to encounter export, to make information about nearby users available to other applications. For instance, exporting a "Nearby" group into the Android address book, which includes the contact details of currently nearby users, if known.

Another feature requested was the ability to create ephemeral pseudonyms as a way to control linkability (in addition to the pairwise linkability offered by EnCore). These pseudonyms allow other users to link their encounters with a device under a pseudonym for a limited period and location (e.g., while attending a conference). Users can change (or remove) their pseudonym and can also choose to stop receiving messages addressed to a pseudonym anytime.

A frequently requested feature is the ability to browse the EnCore timeline, post content and manage events from a desktop computer. Users have also requested the option to thread recurrent or related events, and view posts and comments within a thread in a linearized manner. Finally, users asked that *Context* suggest events and content to post, based on the users' history, preference, and current context. For instance, if Alice, Bob and Charlie have had frequent meetings recently, then *Context* could automatically suggest another instance of the event when it notices similar circumstances (encounters, location, time). These feature requests suggest that users find it useful to be able to explore an encounter timeline, coupled with the ability to create links, and to relate and recount events.

8.2 Risks and challenges

User privacy and security has informed every step of EnCore's design. Unlike virtually all existing mobile social apps, EnCore does not require the user to reveal their sensitive context data, which often combines location, social contact and communication trace, to a provider. Moreover, EnCore prevents Bluetooth device tracking and provides strong security and privacy guarantees within its threat model. However, privacy risks and usability challenges remain.

EnCore database confidentiality The data logged by EnCore resides on the mobile device, and is susceptible to loss, theft, or subpoena. It is not clear what legal rights regarding privacy and self-incrimination, if any, users can assert with respect to data stored on their personal devices. Encrypting the EnCore database protects the data in the case of loss or theft, though it will not stop a court from compelling the user to provide the decryption keys. The risk can be somewhat reduced by configuring the database to store a limited history. Since the usefulness of encounter information likely diminishes over time, the resultant loss of functionality may be acceptable.

Private profile matching Linking encounters based on shared attributes is supported by EnCore, but currently not fully exported by *Context*. A challenge in this regard is how to prevent attacks where a malicious device advertises attributes in order to learn as many attributes of nearby users as possible. The problem can be partly mitigated by ignoring devices that advertise too many attributes or change their attributes too frequently, but a more general defense is hard, unless attributes can be certified by an external authority.

Reliably identifying socially relevant encounters Identifying relevant encounters (e.g., the participants of a shared event) was not a problem in our deployment. The fact that all participants revealed their name or a pseudonym while in the office, combined with the signal strength indication, proved sufficient.

However, identifying socially relevant encounters in a larger and denser environment with many unlinkable devices is an open challenge. For instance, it is important to reliably identify the attendees of a private, closed-door meeting that takes place in an office building with EnCore devices in adjacent rooms. We are currently experimenting with an audio-based confirmation protocol, where devices have to answer a (fast attenuating) challenge transmitted as an audio chirp, in order to identify devices in the same room. Another option would be to follow a non-interactive approach similar to that of Sound of Silence [58], where each device uploads a signature of their acoustic environment to a cloud service that, by comparing signatures, identifies nearby devices.

In other situations, like a crowded party, distinguishing individual attendees is usually not necessary, because the most likely types of interaction (e.g., sharing photos) are directed to the group as a whole. In situations where users wish to identify individuals within a crowded space (e.g., a dinner party at a busy restaurant), people tend to know each other and have their devices linked already. If not, they can resort to bumping devices via NFC or shake-to-connect [28, 50]. In this case, no contact details would be exchanged (unless desired), but the encounter would be marked as "confirmed" on both devices.

Communication with strangers The limited deployment within our institute has not yet allowed us to experiment with communication among strangers, as it would occur, for instance, in the sightseeing scenario described in the introduction. This case, as well as other challenges described above will require experience with deployments at larger scale. Toward this end, we are developing a version of EnCore that does not require rooting the phone, which is currently a major hurdle for a larger deployment. Nevertheless, we believe we have shown that EnCore provides a robust foundation for building secure, privacy-preserving mobile social applications that exploit the opportunities afforded by D2D communication and secure encounters.

9. CONCLUSION AND FUTURE WORK

We have described the design, implementation, and evaluation of EnCore, a mobile platform for social applications based on secure encounters. EnCore can support a wide range of event-based communication primitives for mobile social apps, with strong security and privacy guarantees, without requiring a trusted provider, and while integrating with existing communication, storage and OSN services. As part of our evaluation, we have conducted small-scale deployments of *Context*, an app for event based communication, sharing and collaboration. User experience was favorable: users were engaged, requested new features, and used the app in interesting ways not envisioned by the designers.

While our small-scale deployments have been invaluable in developing the system, secure encounter-based communication promises more than we have been able to evaluate among a small set of mutually trusting, technically savvy users. Evaluating EnCore's primitives in dense environments and among strangers requires larger scale deployment onto a more heterogeneous population. We are in the process of completing a version of EnCore that can be installed on an unrooted phone, which will be much easier to disseminate at scale. Our experience catalogued in this paper gives us confidence that EnCore and the secure encounter primitive will continue to prove useful, and a larger userbase will yield compelling new ways to communicate using EnCore.

10. REFERENCES

[1] AllJoyn. http://www.alljoyn.org. Last accessed: September 2013.

[2] Android Beam. http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#p2p. Last accessed: June 2013.

[3] Bluetooth Specification Core Version 4.0. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737. Last accessed: March 2014.

[4] FireChat. https://itunes.apple.com/us/app/firechat/id719829352?mt=8. Last accessed: March 2014.

[5] Foursquare. https://foursquare.com/. Last accessed: June 2013.

[6] Friday: automated journal. http://www.fridayed.com/. Last accessed: October 2013.

[7] Google fires engineer for violating privacy policies. http://www.physorg.com/news203744839.html. Last accessed: September 2012.

[8] Haggle. http://www.haggleproject.org. Last accessed: September 2013.

[9] Highlight. http://highlig.ht/. Last accessed: December 2013.

[10] iOS 7 AirDrop. http://support.apple.com/kb/HT5887. Last accessed: January 2014.

[11] Lokast. http://www.lokast.com. Last accessed: September 2013.

[12] Mailinator: Free disposable email. http://mailinator.com/. Last accessed: January 2014.

[13] Memoto: automatic lifelogging camera. http://memoto.com/. Last accessed: September 2013.

[14] Near Field Communication – Interface and Protocol (ISO/IEC 18092:2013). http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=56692. Last accessed: September 2013.

[15] Nintendo 3DS. http://www.nintendo.com/3ds. Last accessed: September 2013.

[16] Secret. https://www.secret.ly/. Last accessed: March 2014.

[17] Sony PlayStation Vita. http://us.playstation.com/psvita/. Last accessed: September 2013.

[18] Tile. http://www.thetileapp.com/. Last accessed: September 2013.

[19] Whisper. http://whisper.sh/. Last accessed: March 2014.

[20] Wi-Fi Direct. http://www.wi-fi.org/discover-and-learn/wi-fi-direct. Last accessed: September 2013.

[21] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland. Social fMRI: Investigating and shaping social mechanisms in the real world. *Pervasive Mob. Comput.*, 7(6), Dec. 2011.

[22] W. Apolinarski, M. Handte, M. U. Iqbal, and P. J. Marrón. Secure interaction with piggybacked key-exchange. *Pervasive Mob. Comput.*, 10, Feb. 2014.

[23] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In *Proceedings of the ACM SIGCOMM conference on Data communication*, SIGCOMM '09, 2009.

[24] L. B. Baker and J. Finkle. Sony PlayStation suffers massive data breach. http://www.reuters.com/article/2011/04/26/us-sony-stoldendata-idUSTRE73P6WB20110426. Last accessed: September 2012.

[25] M. Bakht, M. Trower, and R. H. Kravets. Searchlight: won't you be my neighbor? In *Proceedings of the 18th annual international conference on Mobile computing and networking*, MobiCom '12, 2012.

[26] 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*, MobiCom '08, 2008.

[27] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov. "you might also like: " privacy risks of collaborative filtering. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, 2011.

[28] C. Castelluccia and P. Mutaf. Shake them up!: a movement-based pairing protocol for CPU-constrained devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, MobiSys '05, 2005.

[29] 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, 2007.

[30] E. D. Cristofaro, Y. Lu, and G. Tsudik. Efficient techniques for privacy-preserving sharing of sensitive information. Cryptology ePrint Archive, Report 2011/113, 2011. http://eprint.iacr.org/.

[31] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6), nov 1976.

[32] B. Dodson, I. Vo, T. Purtell, A. Cannon, and M. Lam. Musubi: disintermediated interactive social feeds for mobile devices. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, 2012.

[33] P. Dutta and D. Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, 2008.

[34] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris. Persistent personal names for globally connected mobile devices. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, 2006.

[35] M. Goetz and S. Nath. Privacy-aware personalization for mobile advertising. Technical report.

[36] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, 2008.

[37] S. Guha, M. Jain, and V. N. Padmanabhan. Koi: a location-privacy platform for smartphone apps. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, NSDI'12, 2012.

[38] C. A. Gunter, M. J. May, and S. G. Stubblebine. A formal privacy system and its application to location based services. In *Proceedings of the 4th international conference on Privacy Enhancing Technologies*, PET'04, 2005.

[39] B. Han and A. Srinivasan. ediscovery: Energy efficient device discovery for mobile opportunistic communications. In *Proceedings of the 20th IEEE International Conference on Network Protocols (ICNP)*, ICNP '12, 2012.

[40] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, 2008.

[41] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, 2011.

[42] P. Jappinen, I. Laakkonen, V. Latva, and A. Hamalainen. Bluetooth device surveillance and its implications. *WSEAS Transactions on Information Science and Applications*, 1(4), Oct. 2004.

[43] S. Jarecki and N. Saxena. Authenticated key agreement with key re-use in the short authenticated strings model. In *Proceedings of the 7th international conference on Security and cryptography for networks*, SCN'10, 2010.

[44] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Trans. on Knowl. and Data Eng.*, 19(12), Dec. 2007.

[45] A. Kandhalu, K. Lakshmanan, and R. R. Rajkumar. U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, 2010.

[46] V. Lenders, E. Koukoumidis, P. Zhang, and M. Martonosi. Location-based trust for mobile user-generated content: applications, challenges and implementations. In *Proceedings of the 9th workshop on Mobile computing systems and applications*, HotMobile '08, 2008.

[47] M. Lentz, V. Erdelyi, P. Aditya, E. Shi, P. Druschel, and B. Bhattacharjee. SDDR: Light-Weight Cryptographic Discovery for Mobile Encounters. http://www.cs.umd.edu/projects/encore.

[48] 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 *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, 2009.

[49] 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, 2009.

[50] R. Mayrhofer and H. Gellersen. Shake well before use: authentication based on accelerometer data. In *Proceedings of the 5th international conference on Pervasive computing*, PERVASIVE'07, 2007.

[51] J. S. Plank. A tutorial on reed-solomon coding for fault-tolerance in raid-like systems. *Software-Practice & Experience*, 27(9), Sept. 1997.

[52] F. Y. Rashid. Epsilon data breach highlights cloud-computing security concerns. http://www.eweek.com/c/a/Security/Epsilon-Data-Breach-Highlights-Cloud-Computing-Security-Concerns-637161/. Last accessed: September 2012.

[53] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial & Applied Mathematics*, 8(2), jun 1960.

[54] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, HotMobile '09, 2009.

[55] A. Schulman, T. Schmid, P. Dutta, and N. Spring. Demo: Phone power monitoring with BattOr. In *In the 17th ACM international conference on Mobile computing and networking*, MobiCom '11, 2011.

[56] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, 2011.

[57] J. Su, J. Scott, P. Hui, J. Crowcroft, E. De Lara, C. Diot, A. Goel, M. H. Lim, and E. Upton. Haggle: seamless networking for mobile applications. In *Proceedings of the 9th international conference on Ubiquitous computing*, UbiComp '07, 2007.

[58] W.-T. Tan, M. Baker, B. Lee, and R. Samadani. The sound of silence. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, 2013.

[59] K. Thomas. Microsoft cloud data breach heralds things to come. http://www.pcworld.com/article/214775/microsoft_cloud_data_breach_sign_of_future.html. Last accessed: September 2012.

[60] W. Wang, V. Srinivasan, and M. Motani. Adaptive contact probing mechanisms for delay tolerant applications. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, 2007.