SDDR: Light-Weight, Secure Mobile Encounters

Matthew Lentz[†] Viktor Erdélyi[‡] Paarijaat Aditya[‡]
Elaine Shi[†] Bobby Bhattacharjee[†] Peter Druschel[‡]

[†]*University of Maryland* [‡]*MPI-SWS*

Abstract

Emerging mobile social apps use short-range radios to discover nearby devices and users. The device discovery protocol used by these apps must be highly energy-efficient since it runs frequently in the background. Also, a good protocol must enable secure communication (both during and after a period of device co-location), preserve user privacy (users must not be tracked by unauthorized third parties), while providing selective linkability (users can recognize friends when strangers cannot) and efficient silent revocation (users can permanently or temporarily cloak themselves from certain friends, unilaterally and without re-keying their entire friend set).

We introduce SDDR (Secure Device Discovery and Recognition), a protocol that provides *secure encounters* and satisfies all of the privacy requirements while remaining highly energy-efficient. We formally prove the correctness of SDDR, present a prototype implementation over Bluetooth, and show how existing frameworks, such as Haggle, can directly use SDDR. Our results show that the SDDR implementation, run continuously over a day, uses only ~10% of the battery capacity of a typical smartphone. This level of energy consumption is four orders of magnitude more efficient than prior cryptographic protocols with proven security, and one order of magnitude more efficient than prior (unproven) protocols designed specifically for energy-constrained devices.

1 Introduction

Mobile social applications discover nearby users and provide services based on user activity (what the user is doing) and context (who and what is nearby). Services provided include notifications when friends are nearby (Foursquare [6], Google Latitude [7]), deals from nearby stores (Foursquare), content sharing with nearby users (FireChat [5], Whisper [15], Haggle [50]), messaging for missed connections (SMILE [43], SmokeScreen [27]), lost and found (Tile [13], StickNFind [12]), sharing payments with nearby users (Venmo [14]). At their lowest layer, these applications all discover nearby devices; many also associate previously linked users to discovered devices and provide communication among presently or previously co-located devices.

Most commercially deployed solutions rely on a trusted cloud service [6, 7], which tracks users' activity and location, so that it can match co-located users and relay information among them. Discovery using a centralized matchmaking service forces users to disclose their whereabouts, perils of which have been extensively noted [16, 19, 24, 48, 52]. Instead of relying on centralized services, an alternate class of discovery protocols make use of local, short-range radio-to-radio communication [1, 9, 27, 50]. The common practice of using static identifier(s) in the discovery process [2] leaks information, since it allows an eavesdropper to track a user's locations and movements. To protect against such tracking, previous work [35–37] has suggested that ephemeral identifiers should be used in place of static ones. Simply replacing static identifiers with strictly random ephemeral identifiers is insufficient: while eliminating tracking, it also prevents friends (or users with prior trust relations) from recognizing each other when nearby.

In this paper, we describe a light-weight, energy-efficient cryptographic protocol for *secure encounters* called SDDR. At a high level, secure encounters provide the following properties: 1) discovering nearby devices, 2) mapping devices to known principals (if possible), and 3) enabling secure communication for encounter peers.

Device discovery and secure encounter SDDR performs a pair-wise exchange of a secret with each nearby device. The shared secret enables encounter peers to communicate securely during and after the encounter, anonymously and without trusting a third party (e.g., sharing related content with event participants).

Selective linkability and revocation Additionally, SDDR enables a user's device to be identifiable by specific other users, while revealing no linkable information to other devices. For instance, friends can agree to recog-

nize each others' devices, while third parties are unable to link and track devices upon repeat encounters. Moreover, users can efficiently and unilaterally revoke or suspend this linkability, for instance based on the current time or location (e.g., discoverable by colleagues only during work hours and on company premises).

Challenges: Energy efficiency and DoS resilience In theory, designing a protocol that satisfies the above functional and security requirements is straightforward. For example, an inefficient strawman scheme can be constructed using existing cryptographic primitives. Pairs of devices can perform a Diffie-Hellman key exchange to establish a shared secret, enabling the users to securely communicate. To support selective linkability, two users can participate in a standard Private Set Intersection (PSI) protocol. A user can allow (or disallow) a peer to recognize them in a future encounter by including (or excluding) a past shared encounter secret from the set.

However, as we will show in Section 6, using a full-fledged PSI protocol is impractical. Because the shared encounter secrets (i.e., elements in the set) are high-entropy values, it is possible to implement a secure PSI protocol through an efficient Bloom filter based construction. Unfortunately, even when using an efficient Bloom filter based PSI scheme, the above strawman scheme—implemented naively—has high energy consumption. Specifically, a naive implementation requires a device to wake up its CPU each time it receives a message from a nearby device, an expensive operation for energy-constrained mobile devices. The protocol would deplete the battery in crowded spaces (e.g., a subway train) where hundreds of devices may be within radio range. Furthermore, an attacker mounting a DoS attack could deplete the victim device's battery by frequently injecting messages to cause unnecessary wake ups.

1.1 Contributions

We designed, implemented, and formally proved the security of SDDR, a light-weight secure encounter protocol suitable for resource-constrained mobile devices. Our reference implementation source code (using Bluetooth 2.1 as the short-range radio) is available at http://www.cs.umd.edu/projects/ebn.

Achieving energy efficiency The main feature of SDDR is its *non-interactiveness*, i.e., the encounter protocol consists of periodic broadcasts of beacon messages, which enable both the key exchange and selective recognition. Because the SDDR protocol is non-interactive, the Bluetooth controller can be initialized so that it responds to discovery requests from peers with a beacon message, while the main CPU remains completely in the *idle state*. A device only needs to wake up its CPU when actively discovering nearby peers.

Our evaluation shows that such a non-interactive protocol allows us to improve the energy efficiency by at least 4 times in comparison with any interactive protocol (even if the interactive protocol performs no work), under a typical setting with 5 *new* devices nearby on average during *every* 60 second discovery interval. Under the same parameters, we show that an otherwise idle device running SDDR over Bluetooth 2.1 will operate for 9.3 *days* on a single charge.

First formal treatment of the problem We are the *first* (to the best of our knowledge) to provide a formal treatment of secure device discovery and recognition. We define a security model that captures the requirements of secure encounters and selective linkability, and prove that our solution is secure under the random oracle model (see Appendix A.3).

Applications over SDDR To demonstrate some of SDDR's capabilities, we have modified the Haggle mobile networking platform to use SDDR, enabling efficient and secure discovery and communication via Bluetooth for all Haggle apps. For demonstration, we have modified the PhotoShare app to enable private photo sharing among friends using SDDR selective linking.

Roadmap The remainder of the paper is organized as follows. We discuss related work in Section 2. Next, we review security requirements, formulate the problem and provide security definitions in Section 3. We present details of the SDDR discovery protocol in Section 4, followed by our reference Bluetooth implementation and evaluation results in Sections 5 and 6, respectively. We discuss the properties and implications of SDDR's encounter model in Section 7. We conclude in Section 8.

2 Related Work

Device discovery protocols Several device discovery protocols have been proposed; however, none simultaneously offer the full functionality and security offered by our SDDR protocol. Since SDDR provides secure device discovery and recognition for a large range of mobile encounter applications, it allows developers to focus on their application logic.

	Unlinkability	Selective Linkability	Efficient Revocability	No Trusted Party	Record Encounters
Bluetooth 4.0	✓	✓		✓	
SMILE	✓			✓	✓
SmokeScreen	✓	✓			✓
SDDR	✓	✓	✓	✓	✓

Table 1: Comparison of related device discovery and recognition protocols in terms of supported properties.

Bluetooth 4.0 (BT4) is the most recent version of the Bluetooth standard, introducing a new low-energy mode [3], as well as support for random MAC addresses to be used in communication. Building on top of the MAC address change support, BT4 adds a form of selective linkability in which paired (trusted) devices can recognize each other across MAC address changes, while remaining unlinkable to all other devices. Since BT4 uses a single shared key for all currently linkable users, it does not allow for efficient revocation of a subset of users. Further, BT4 does not natively support encounters with unlinkable devices.

SMILE [43] is a mobile "missed connections" application, which enables users to contact people they previously met, but for who they don't have contact information. The SMILE protocol creates an identifier and shared key with 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 identifier. Unlike SDDR, SMILE does not address selective linkability and revocation.

MeetUp [44] is an encounter-based social networking application that argues for (and uses) strong authentication within an encounter. This authentication comes in the form of exchanging signed certificates (from a trusted authority) attesting to a public key and picture of a user. However, we feel that in many applications, users should be unlinkable by default, and should not be required to distribute any identifiable information (e.g., public key, user picture) in an encounter. We discuss authentication in Section 4.4.

SmokeScreen [27], a system that allows friends to share presence while ensuring privacy, also implements a selectively linkable discovery protocol for encounter peers. In SmokeScreen's discovery protocol, devices periodically broadcast two types of messages: *clique signals* and *opaque identifiers*. Clique signals enable private presence sharing among friends, announcing the device's presence to all members of a mutually trusting clique. In comparison with SDDR, SmokeScreen requires a trusted third-party service and achieves slightly weaker security: an adversary can infer that two users belong to the same clique, since all users broadcast the same clique signal during each time epoch. Furthermore, SDDR can handle 35 nearby devices for the same energy as 3 devices in SmokeScreen. Additionally, SDDR supports efficient revocation of linked users, which is not possible with cliques in SmokeScreen.

SlyFi [35] is a link layer protocol for 802.11 networks that obfuscates MAC addresses and other information to prevent tracking by third parties. Unlike SDDR, SlyFi does not address selective linkability or revocation, nor does it negotiate shared keys among co-located devices. SDDR includes a Bluetooth MAC address change protocol similar to SlyFi's to prevent tracking.

Related protocols using Bloom filters Bloom filters [20] are a space-efficient probabilistic data structure for set membership. Bloom filters have been used in many cryptographic protocols [23], including (private) set-intersection and secure indexes. However, none of the protocols address the precise problem and security requirements of SDDR.

Secure indexes are data structures that allow queriers to perform membership tests for a given word in $O(1)$ time if they have knowledge of the associated secret. Secure indexes were first defined and formalized by Goh [33], who provided a practical implementation using Bloom filters. Similar work has focused on privacy-preserving searches over encrypted data [26] and databases [54] using Bloom filters. If applied to device recognition, all protocols would allow adversaries to track users due to the static Bloom filter content.

PrudentExposure [56] allows users to privately discover appropriate services, where the user and service belong to the same domain. To maintain user privacy, PrudentExposure relies on Bloom filters containing time-varying hashes of domain identities for intersecting the requested and available domains.

E-SmallTalker [55] and D-Card [25], which builds on E-SmallTalker, support social networking with nearby strangers (E-SmallTalker) and friends (D-Card). BCE [31] enables users to estimate the set of common friends with other users. These protocols would be insecure when applied to the device recognition problem, as none of the protocols use time-varying information in the Bloom filters, allowing users to be linked across multiple handshakes. Additionally, E-SmallTalker does not apply the Bloom Filter to high-entropy secrets, and thus is vulnerable to an offline dictionary attack.

Sun et al. [51] present a new way of building trust relationships between users by comparing spatiotemporal profiles (log of time and location pairs). In addition to a PSI-based scheme, they present another scheme using Bloom filters that trades off estimation accuracy and privacy in a user-defined manner. In SDDR, we avoid the privacy vs. accuracy trade off since the linkable users share a high-entropy secret as opposed to low-entropy time, location pairs.

Dong et al. [30] use *garbled* Bloom filters to create a practical PSI protocol that handles billions of set members. While more efficient than existing PSI protocols, it does not *scale down* when applied to small set sizes on resource-constrained devices. Because of its reliance on secret shares instead of bits in the Bloom filter, the smallest possible Bloom filter to handle a maximum of 256 items would be 17736 bytes — two orders of magnitude larger than what SDDR requires. In addition, the

communication cost of this interactive protocol increases linearly with the number of nearby devices.

Nagy et al. [45] use Bloom filters to provide a PSI protocol that allows users of online social networks (OSNs) to determine common friends while preserving user privacy. While their solution provides ample efficiency gains over standard PSI, saving an order of magnitude in communication and computation costs, several seconds per *interactive* exchange is too much when running on power-constrained devices in dense environments.

Authenticated key exchange Secure device discovery and recognition should not be confused with mutual authentication, or authenticated key exchange (AKE) protocols [21, 40]. SDDR aims to achieve device discovery and recognition; guaranteeing mutual authentication is not a goal of the basic SDDR protocol. As noted in Section 4.4, once Alice's device recognizes Bob's device, Alice can authenticate Bob by soliciting an explicit verification message from Bob; however, authentication will only be performed if desired by the higher-level application (or user). While secure device discovery and recognition can be achieved by executing an AKE protocol with each nearby device (for all possible shared secrets), such a scheme would be prohibitive in an environment with many nearby peers.

3 Problem Overview

In this section, we review the requirements for a secure encounter protocol, sketch a strawman design, and make observations that enable a practical protocol.

Devices executing a *secure encounter protocol* should detect nearby participating devices, and learn their current ephemeral network identifier. Additionally, each pair of nearby devices should generate a unique (except with negligible probability) shared secret key, known only to the pair. This key allows the devices to: 1) uniquely refer to a particular encounter; and, 2) authenticate each other as the peer in the encounter and securely communicate. The pair should learn no other information about each other; when the same pair of devices meet again, the shared secret and network identifiers exchanged should be unrelated.

By default, devices should remain unlinkable, meaning that no identifying information is exchanged. While unlinkability is appropriate between strangers' devices, friends may wish to enable their devices to recognize each other. A user who allows her device to be recognized by a friend during future discoveries is termed *selectively linkable* (or simply *linkable*) by that friend. When two devices discover each other, a *recognition protocol* should determine if the remote device corresponds to a linkable user. Selectively linkable users must share a unique secret value such that the devices can authenticate each other during the recognition protocol; we refer to this shared secret as the *link value*. Users can derive the link value from the shared secret established during a prior encounter, or using an out-of-band protocol.

In general, users may not wish to be recognizable by their entire set of friends at all times (e.g., Alice may only want her work colleagues to recognize her device while at work). Therefore, a user should be able to contextually (e.g., in terms of time, place, activity) filter the set of friends that can recognize them. This filtering requires that revocation of selective linkability be efficient (e.g., not require a group re-keying) and unilateral (e.g., not require communication). Additionally, the filtering may take place in one direction: Alice may want to not be recognizable by Bob, yet still want to recognize him. Therefore, we consider two distinct sets of link value: the set of advertiseIDs (i.e., who you are willing to be recognized by), and the set of listenIDs (i.e., who you want to recognize). Alice's device is able to recognize Bob's device if and only if their shared link value is in Bob's advertiseIDs and in Alice's listenIDs.

3.1 Security Requirements

We summarize the security requirements below:

Secure communication If Alice and Bob share an encounter, they are able to securely communicate using an untrusted communication channel, both during and after the encounter, and regardless of whether Alice and Bob have opted to selectively link their devices.

Unlinkability The information exchanged during a secure encounter reveals no identifying information about the participating devices, unless the devices have been explicitly linked. In particular, unlinked devices that encounter each other repeatedly are unable to associate their encounters with a previous encounter.

Selective linkability Alice and Bob can optionally agree to be linkable, and therefore able to recognize and authenticate each others' devices in subsequent discoveries.

Revocability Alice may, at any time, *unilaterally* revoke Bob's ability to recognize her.

3.2 Threat Model

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. Preventing such leaks is an orthogonal concern outside SDDR's threat model. User devices attempt to participate in the protocol with all nearby discovered devices, a subset of which could be controlled by attackers, who may all collude.

We do not consider radio fingerprinting attacks, which detect a device by its unique RF signature [22]. Such

attacks may require sophisticated radio hardware, and are outside our threat model.

3.3 Strawman Protocol

A strawman scheme using existing cryptographic tools, namely Diffie-Hellman [29] (DH) and Private Set Intersection [28, 39] (PSI), can meet the requisite security requirements outlined above. Upon detecting a device, the protocol performs a DH exchange to agree upon a shared secret key. By generating a new DH public and private key pair prior to each exchange, devices remain *unlinkable* across encounters.

To recognize selectively linkable devices, the protocol executes PSI over the devices' advertised and listen identifier sets. *Selective linkability* and *revocability* properties are satisfied by all PSI protocols; however, in order to preserve privacy, we require a PSI protocol that supports *unlinkability* across multiple executions.

While the DH+PSI strawman achieves the desired security properties, it is not practical when frequently run on resource constrained devices. As shown in Section 6, the computation and communication requirements of existing PSI constructions are prohibitively high.

3.4 Observations

In order to enable a practical protocol we rely on several observations:

First, *strict unlinkability* requires that two different discoveries between a pair of devices are unlinkable, regardless of how closely the discoveries are spaced in time. This property cannot be achieved with a non-interactive protocol, because it requires a change of ephemeral network ID and DH keys after each discovery. In order to use a non-interactive protocol, we must settle for the slightly weaker property of *long-term unlinkability*; devices may be linkable within a time epoch, but they remain unlinkable across epochs. For an epoch on the order of minutes, long-term unlinkability is sufficient in practice. It is important to note that epoch boundaries and durations do not require time synchronization; devices may choose when to change epochs independently.

Second, detecting selectively linked devices requires an intersection of the sets of advertised link values between a pair of devices. Even a simple, *insecure* intersection protocol would require the transmission of the complete sets during each pair-wise device discovery, which is too expensive. However, we note that in a large deployment, discoveries among strangers are far more common than discoveries among linked devices. Therefore, an over-approximation of the set intersection may suffice. False positives can be resolved when two presumed linkable devices attempt to authenticate each other using the shared link value.

Finally, we can take advantage of the fact that link values shared between users are high-entropy values taken from a large space, by design. General purpose PSI protocols, on the other hand, ensure security even when sets contain low-entropy values (e.g., dictionary words).

Using these observations, we present the SDDR protocol, which meets the security requirements with practical performance and energy efficiency.

4 SDDR Design

4.1 High-Level Protocol

Like the strawman protocol, SDDR uses DH to exchange a shared secret key with each nearby device; however, SDDR performs the exchange in a non-interactive manner. Periodically, each device broadcasts its DH public key and receives broadcasts from other nearby devices, computing all pair-wise shared secret keys.

SDDR divides time into epochs, during which the ephemeral network address, DH public/private key pair, and advertiseIDs set digest remain constant. Devices are unlinkable *across* epochs, thus preserving *long-term unlinkability*. To avoid expensive synchronous communication, epochs are not synchronized among devices. As a result, the DH computation may fail to produce a shared key if it occurs around an epoch change of either device in a pair. For instance, Alice receives Bob's broadcast in her epoch n, but Bob fails to receive Alice's broadcast until her epoch $n+1$, so he computes a different key. Because broadcasts occur more frequently than epoch changes (seconds versus minutes), however, the probability that a broadcast round yields a *shared* key quickly tends to one with every broadcast round.

Since the link identifiers shared between users are high-entropy values chosen from a large space (e.g., a shared key produced during a prior discovery), SDDR can recognize linkable devices by broadcasting salted hashes of their respective set of advertiseIDs. The DH public key is used as the salt; since the salt is different in each epoch, a device cannot be recognized by the bit-pattern in its Bloom filter across epochs, that ensuring long-term unlinkability. Each user then searches over the hashes using their own set of listenIDs, along with the corresponding salt value, in order to identify the listenID (if any) associated with the remote device.

However, the communication required for moderately-size sets (e.g., 256 advertiseIDs) is still too large for an efficient implementation in Bluetooth due to (pseudo-) broadcast message length constraints. By allowing the recognition protocol to *over-approximate* the actual intersection between the set of local listenIDs and remote advertiseIDs, SDDR can use a probabilistic set digest data structure to reduce the communication needed to determine the intersection. The size of the set digests can be parameterized based on the message size restrictions of

the radio standard used for communication. The choice affects performance only; false positives due to the use of set digests can be resolved using the shared link values, and therefore have no bearing on the protocol's security.

The *selective linkability* property is satisfied by the use of non-deterministic hashes of the link identifiers shared by two users, only allowing linkable users to recognize each other. The *revocation* is supported by the user's ability to add or remove link values from the set of advertiseIDs.

4.2 Formal Problem Definition

We divide the non-interactive SDDR protocol into two algorithms (GenBeacon and Recognize), which we formalize below:

$\mathsf{beacon} \leftarrow \mathsf{GenBeacon}(\mathsf{advertiseIDs})$

In each epoch, a device wishing to participate in peer encounters executes the GenBeacon algorithm, which takes as input the current set of advertiseIDs. The GenBeacon protocol outputs a message beacon, which the device then broadcasts to nearby devices.

$(sk, \mathsf{listenIDs_{re}}, \mathsf{L}) \leftarrow \mathsf{Recognize}(\mathsf{beacon_{re}}, \mathsf{listenIDs})$

Upon receiving a $\mathsf{beacon_{re}}$ from a remote peer, a device executes the Recognize algorithm, which additionally takes in the current set of listenIDs. The Recognize algorithm outputs a secret key *sk*, the set of $\mathsf{listenIDs_{re}}$ associated with the remote peer, and the link identifier L for this encounter.

4.3 Detailed Protocol Description

Next, we provide a detailed description of the SDDR protocol. Pseudo-code for the *GenBeacon* and *Recognize* algorithms is shown in Figure 1. In the protocol, as well as our implementation, we use Bloom filters as the probabilistic set digest data structure; however, other set digests (e.g. Matrix filters [46]) could be used instead.

Each user P_i starts by running GenBeacon in order to generate the beacon message to broadcast during the current epoch. GenBeacon first selects a random DH private key α_i, which corresponds to the DH public key g^{α_i}. Afterwards, GenBeacon computes the Bloom filter by hashing each advertiseID within AS_i (the set of advertiseIDs), using the DH public key as the salt. The resulting beacon contains the public key and the Bloom filter.

Each user P_i broadcasts their respective beacon during the epoch. After receiving a beacon from a remote user P_j, user P_i runs the Recognize algorithm. Recognize first computes the DH secret key dhk_{ij}, using the local user's DH private key and the remote user's DH public key (as contained in the beacon). Using the dhk_{ij} along with the local and remote DH public keys, Recognize computes the shared link identifier L_{ij}, which can optionally be used in case the two users wish to selectively link.

Additionally, Recognize computes the key sk_{ij} using the link identifier L_{ij}, which the two devices can use to authenticate each other as the peer associated with this encounter, and then securely communicate. Finally, Recognize queries the Bloom filter by hashing each listenID within LS_i (the set of listenIDs), using the remote user's DH public key as the salt, resulting in the set of matches M_j.

Recall that sk_{ij} may not be shared (i.e., $sk_{ij} \neq sk_{ji}$) in some cases when individual devices decide to change epochs. When a device attempts to communicate using such a key, the authentication will fail, and the device retries with a key produced in a subsequent discovery. Also, to make sure a valid link identifier is used, devices attempt to authenticate each other as part of the pairing process to selectively link.

Notation: Let $\mathsf{BF}\{S\}$ denote a Bloom filter encoding the set S. Let H_0, H_1, and H_2 denote independent hash functions later modeled as random oracles in the proof.

Inputs: Each user P_i has a set of listenIDs (LS_i) and a set of advertiseIDs (AS_i).

Outputs: For all users P_j, discovered by P_i, P_i outputs:

1. sk_{ij}: A shared secret key
2. $\mathsf{listenIDs_{re}}$: Set of matching $\mathsf{listenID} \in LS_i$ associated with P_j
3. L_{ij}: A shared link identifier

Protocol: Each P_i performs the following steps:

$\underline{\mathsf{GenBeacon}(AS_i)}$

1. Select random $\alpha_i \in_R \mathbb{Z}_p$
2. Compute $\mathsf{BF}_i := \mathsf{BF}\{H_1(g^{\alpha_i}||x) : x \in AS_i\}$
3. Create $\mathsf{beacon}_i = (g^{\alpha_i}, \mathsf{BF}_i)$

Each user P_i broadcasts beacon_i***. For each user P_j that user P_i discovers, P_i runs*** Recognize**.**

$\underline{\mathsf{Recognize}(\mathsf{beacon}_j, LS_i)}$

1. Compute DH key $\mathsf{dhk}_{ij} = (g^{\alpha_j})^{\alpha_i}$
2. Compute link $\mathsf{L}_{ij} := \begin{cases} H_0(g^{\alpha_i}||g^{\alpha_j}||\mathsf{dhk}_{ij}) \text{ if } g^{\alpha_i} < g^{\alpha_j} \\ H_0(g^{\alpha_j}||g^{\alpha_i}||\mathsf{dhk}_{ij}) \text{ otherwise} \end{cases}$
3. Compute key $sk_{ij} := H_2(\mathsf{L}_{ij})$
4. Query for set $\mathsf{M}_j := \{x : x \in LS_i \wedge H_1(g^{\alpha_j}||x) \in \mathsf{BF}_j\}$

Figure 1: SDDR Non-Interactive Protocol.

Hiding Bloom filter load After receiving multiple Bloom filters, and calculating the distribution of the number of bits set, it is possible to determine the size of the remote user's set of advertiseIDs. This leaks information, which could be used to link devices across multiple epochs. To prevent this leak, the Bloom filters are padded to a global, uniform target number of elements N. Rather than computing actual hashes, we randomly

select $K * (N - |\mathsf{advertiseIDs}|)$ (not necessarily distinct) bits to set to 1, where K is the number of hash functions used in creating the Bloom filter.

4.4 Identification and Authentication

Identification and mutual authentication are not required by all applications, and hence are not a part of the basic SDDR protocol. However, identification and authentication can be achieved easily on top of SDDR as follows:

Identification Identification allows a user to associate a principal (e.g., "Bob") to a specific encounter through the use of out-of-band (OOB) mechanisms. As part of the identification procedure, the users agree on the link identifier (corresponding to the shared encounter) for the purpose of selective linkability. If Alice wishes to be recognizable by Bob in the future, she will insert the link identifier into her advertiseIDs; likewise, if she wishes to recognize Bob in the future, she will insert the link identifier into her listenIDs. However, choosing to enable (or revoke) recognizability is not part of the identification procedure, and can be performed any time by the user after the initial, one-time identification has taken place.

It is well known that achieving secure identification, resistant to man-in-the-middle (MITM) attacks, requires either an *a priori* shared secret or an OOB channel. Any manual authentication technique [32,41,42,53] (e.g., displaying and comparing pictures on both devices, generated from the link identifiers) allows Alice to securely identify Bob's device free of MITM attacks. Additionally, a technique not relying on OOB mechanisms has been proposed by Gollakota et al. [34] for 802.11, using tamper-evident messaging to detect and avoid MITM attacks. Note that many applications do not require identification, such as when users wish to (anonymously) share photos with other event participants.

Mutual authentication Mutual authentication bootstraps a secure and authenticated session between two peers using an *a priori* shared secret (e.g., the link identifier agreed upon as part of the identification procedure). Suppose that in a previous encounter, Alice and Bob participated in the identification procedure; additionally, both Alice and Bob elected to add the shared link identifier to both their advertiseIDs and listenIDs. Thus, in future encounters, Alice and Bob can now authenticate each other (free of MITM attacks). While the basic SDDR protocol does not provide authentication, it can easily be achieved by sending an explicit verification message. For example, a user can prove to a remote peer that they know the common link identifier L by simply sending the verification message $\langle \text{nonce}, H_3(L||\text{nonce}||\text{dhk})\rangle$.

Alternatively, a user can execute a standard authenticated key exchange (AKE) protocol; however, in the case of SDDR, since a DH key is already exchanged, an explicit verification message is sufficient and cheaper than a standard AKE protocol. Mutual authentication only needs to be performed when requested by an application (or user), and thus is not part of the base SDDR protocol.

4.5 Suppressing Bloom filter false positives

The false positive probability of a Bloom filter, denoted as P_{fp}, is computed as a function of: the number of elements inserted (N), the size (in bits) of the Bloom filter (M), and the number of hash functions per element (K). Although Bose et al. [47] provide a more accurate (yet not closed form) solution, P_{fp} is closely approximated by the following formula:

$$P_{\mathsf{fp}} = \left(1 - \left[1 - \frac{1}{M}\right]^{KN}\right)^{K}$$

In the SDDR protocol, these false positives manifest themselves as selectively linkable principals associated with the remote device (and their current shared encounter). By default, false positives are not reduced over the course of an epoch, and only mutual authentication (see Section 4.4) will allow two peers to check if they are selectively linkable (resolving any false positives). Ideally, we want to provide a way for the matching set (M_j in the protocol) to converge towards the exact intersection of the remote peer's advertiseIDs and the user's listenIDs over time.

If one creates multiple Bloom filters, each using a unique set of hash functions (or salt value(s)), the intersection of the matching sets for the Bloom filters results in an overall matching set with a reduced false positive rate. Within a single epoch, a device can compute and distribute beacons with unique Bloom filters that evolve over time. Since beacons within the same epoch use the same DH public key, we modify Step 2 within the GenBeacon algorithm to additionally use an incrementing counter *count* as part of the salt:

$$\mathsf{BF}_i := \mathsf{BF}\{H_1(g^{\alpha_i}||x||count) : x \in AS_i\}$$

count increments each time a beacon is broadcast, and is reset to 0 at the start of every epoch. We use this extension as part of our implementation, as described in Section 5.

5 SDDR on Android

We have implemented the SDDR protocol on the Android platform as part of a system service. The codebase is written in C++ and runs with root privileges[1] . For development and evaluation, we use Samsung Galaxy

[1] SDDR requires root privileges to handle address changes, as well as to support efficient communication through Extended Inquiry Response (EIR) payloads.

Nexus phones running Android 4.1.2 [2] with the android-omap-tuna-3.0-jb-mr0 kernel. For our implementation, we selected to use Bluetooth for short-range radio communication; other short-range radios (e.g. WiFi, ZigBee) could also be used for this purpose. We selected Bluetooth 2.1 (BT2.1) over BT4 because a BT2.1 implementation closely mirrors the protocol as described; however, we designed a BT4 implementation for use in EnCore [17]. We use elliptic curve cryptography (ECC) due to the smaller key sizes relative to RSA, selecting the 192-bit curve as recommended by NIST [4].

We first describe the implementation of the major components of the SDDR protocol: discovery, handshake, and epoch change. Afterwards, we briefly describe the system service that we developed to allow all applications running on the device to take advantage of SDDR, without each independently managing discoveries. Finally, we discuss our integration of the system service with the open-source Haggle framework [8].

5.1 SDDR Protocol Components

Discovery In the protocol, a single beacon is broadcast throughout each epoch. In our implementation, since devices must wake up to discover nearby devices and receive their beacon messages, we break down each epoch into multiple discovery intervals. Using the protocol extension described in Section 4.5 to reduce Bloom filter false positives, we generate and broadcast a new beacon during every discovery interval.

There are two roles that devices can take on within the Bluetooth 2.1 discovery protocol: *discoverable* and *inquirer*. Every device always plays the role of *discoverable*, responding to incoming inquiry scans with information on how the *inquirer* can establish a connection (e.g., MAC address). By using the extended inquiry response (EIR) feature present in BT2.1, which includes an additional 240 byte payload added to the response, *discoverable* devices can transmit their beacon to the discoverer during the inquiry scan itself.

At the start of each discovery interval, a device additionally takes on the role of *inquirer*, performing an inquiry scan in order to collect and process beacon messages from nearby devices. In addition, the device will update its EIR payload with a new beacon message; this payload will be used as a response while *discoverable*.

Devices must only wake up when acting as a *inquirer*. Otherwise, while simply *discoverable*, only the Bluetooth controller (and not the main CPU) must be active; the controller wakes up every 1.28 seconds to listen and respond to inquiry scans from nearby devices.

[2]The Android 4.4 release would provide additional energy savings with respect to suspend and wakeup transitions due to the updated AlarmManager API.

Compute keys and recognize When a *inquirer* detects a new device, which could also be an epoch change by an existing device, it computes the shared secret for the current epoch using the local DH private key and the remote device's DH public key (embedded in the beacon). For each device: 1) for its first beacon, the *inquirer* queries the Bloom filter contained in the beacon using $H_1(g^{\alpha_i}||x||count)$ for each x in its set of listenIDs; 2) for subsequent beacons, the *inquirer* queries the Bloom filter only for each x previously determined to be in the intersection.

Periodic MAC address change SDDR ensures that the discovery and recognition protocol does not leak linkable information. However, the underlying Bluetooth packets have a static MAC address that can be used to track the device (and the user). As part of our Bluetooth implementation, we choose a random Bluetooth MAC address at the start of every epoch. BT2.1 does not provide a native interface for changing MAC addresses "on-the-fly"; therefore, we reset the Bluetooth controller each time the address is changed (once per epoch, nominally fifteen minutes). Unfortunately, this reset closes ongoing connections and invalidates existing device pairings; however, the BT4 specification supports changing the public (random) address for the device while maintaining the private address for paired devices.

5.2 SDDR Integration

We chose to implement the SDDR protocol as part of a system service on Android. The centralized service allows for greater energy efficiency as it can broadcasts discovery information to all applications via IPC mechanisms, as opposed to each application performing its own discovery. Currently, we allow applications to connect to the service via local Unix sockets. Applications receive messages for each discovered device, along with the shared secret and identity information (if selectively linkable) for SDDR-aware devices.

Haggle Haggle is a mobile communication platform for device-to-device radio communication, and supports a number of content sharing apps. A photo sharing app, for instance, shares with nearby devices photos whose textual attributes match a user's specified interests.

To demonstrate some of SDDR's capabilities, we have modified Haggle to use SDDR. This enables Haggle and its applications to communicate securely with nearby devices, without revealing any linkable information and without the risk of tracking by third parties. We have modified Haggle's photo sharing app to take advantage of SDDR's features. Users can add a special attribute to a photo, which narrows its visibility to a specified set of linkable user(s). If a photo carries this attribute, it is eligible for sharing only with devices of linkable users in

this set. Finally, when a photo is shared, it is encrypted with the shared key established by SDDR.

6 Experimental Results

In order to evaluate the design and implementation of SDDR, we first perform a comparison to the strawman protocol by focusing on the PSI portion in comparison to our Bloom filter-based approach. Secondly, we look at energy consumption both at the level of benchmarking individual operations within the SDDR (and other) protocols, as well as battery life consumption over the course of a day. Additionally, we attempt to analyze the scalability (and DoS resistance) of the SDDR protocol by extrapolating energy consumption results to a large number of devices. For a more application-level evaluation, we refer the reader to our work on EnCore [17], which includes a deployment with 35 users, using a BT4 implementation of SDDR.

6.1 Comparison with PSI

We measure the SDDR discovery protocol computation time while varying the number of linkable identifiers, and compare the elapsed time to that required for a PSI protocol. We use an implementation [11] of the JL10 scheme [38], one of the fastest schemes known-to-date. JL10 is secure and can be modified to achieve unlinkability across sessions. Both protocols are executed using a single core on the 1.2 GHz ARM Cortex-A9 processor.

Figure 2 shows the run times for each protocol; each bar is an average of 50 runs, with error bars denoting the 5^{th} and 95^{th} percentile values. We divide SDDR into two separate trials, varying the number of advertisements in order to achieve the specified false positive rates for each trial. Additional advertisements do not require much computation time because SDDR only uses the complete set of listen IDs for the first Bloom filter; afterwards, SDDR uses the matching set of listen IDs, which quickly converges towards the actual intersection.

Results show that SDDR is up to *four orders of magnitude* faster than standard PSI. The gain in performance is crucial for practical deployment, as these computations take place for every discovered device. In order to preserve user privacy against tracking, large maximum set sizes (128 to 256 entries) with random entry padding must be used with typical PSI protocols; alternatively, a size-hiding PSI scheme [18] can be used, but the performance in practice is worse than the scheme we used [10].

6.2 Energy Consumption

SDDR runs on resource-constrained devices, therefore we evaluate its energy consumption in detail. First, we look at microbenchmarks for the individual components of the protocol (e.g., discovery), as well as various idle states, which provide a baseline for energy consumption.

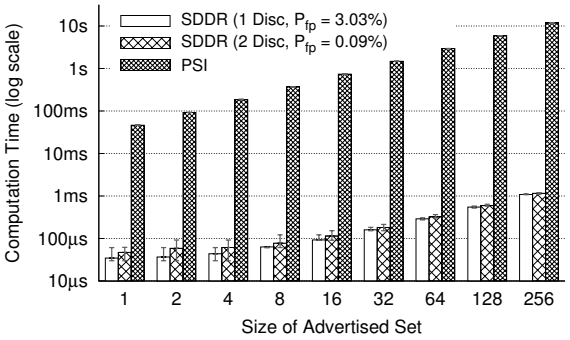

Figure 2: Protocol execution times of PSI versus SDDR for an encounter with varying sizes of advertised sets of link values. The "# disc" represents the number of discovery beacons used to compute the matching set, with P_{fp} as the associated false positive probability.

Second, we collect and analyze power traces of our protocol over several epochs in order to determine the battery life cost of frequently running our handshake protocol over the course of a day. Third, we estimate the reduction in battery consumption when in densely crowded areas, or under denial of service attacks, a device discovers the specification maximum of 255 devices per inquiry scan [3]. In order to monitor energy consumption over time, we use the BattOr [49] power monitor.

Microbenchmarks In Table 2, we outline the results from the microbenchmark experiments. We collect 25 data points for each experiment, and present the average values in the table above. We enable airplane mode on the device for each test, ensuring that all radio interfaces are disabled unless otherwise explicitly requested. Idle state requires very little power, as the device remains in suspended state with the main processor powered off. Since epoch changes require disabling and re-enabling the hardware Bluetooth controller, the controller requires several seconds to return to its working state. An epoch change requires 568mJ energy consumption — however, note that epoch changes are relatively infrequent (e.g., every fifteen minutes) compared to discovery.

Additionally, we collected power traces for various discovery and recognition protocols. When there are no nearby devices, the baseline discovery protocol in Bluetooth 2.1 costs 1363mJ per discovery. In comparison, our implementation of the SDDR protocol over Bluetooth 2.1 incurs only 7% additional energy cost while executing the recognition protocol with 5 nearby devices (and using 256 advertised and listen IDs). The implementation of the DH+PSI strawman over Bluetooth 2.1 requires much more energy per execution, over an order of magnitude greater than the baseline (43,335mJ compared to 1,363mJ). This is expected as the PSI protocol

Component	Avg. Power (mW)	Energy (mJ)
Idle	1.73	-
Bluetooth 2.1		
Discovery (0 Devices)	118	1,363
Incoming Connection	200	893
Discovery and Recognition (5 Devices, 256 Listen IDs)		
SDDR over BT2.1	124	1,464
DH+PSI over BT2.1	404	43,335
ResolveAddr in BT4.0	226	737
SDDR Epoch Change	178	568

Table 2: Average power and energy consumed by various components, or system states. Components which have energy consumption marked as '-' have no well defined duration.

State	Energy (J)	Battery (%)
Full Battery	23,328	100
Idle	150	0.64
Idle with Bluetooth	188	0.81
Running (5 Devices, 256 Listen IDs) (60s Discovery Interval)		
SDDR over BT2.1	2,511	10.76
ResolveAddr in BT4.0	1,260	5.40
DH+PSI over BT2.1	44,619	191.27
IncConn over BT2.1	9,143	39.19
(120s Discovery Interval)		
SDDR over BT2.1	1,300	5.57
ResolveAddr in BT4.0	718	3.08
DH+PSI over BT2.1	35,097	150.45

Table 3: Energy and battery life consumption for different states and protocol configurations over the course of one full day. A daily battery consumption of $p\%$ means that the battery would last $100/p$ days if the device runs the corresponding protocol and is otherwise idle.

is not as efficient as SDDR in terms of computation (See Figure 2) and communication, and it must execute an interactive protocol for each nearby device.

The ResolveAddr protocol, implemented as per the Bluetooth 4.0 specification, requires less energy (737mJ) compared to other schemes; however, it neither exchanges a session key, nor supports efficient revocation of the set of linked users [3]. ResolveAddr is optimized to support a limited feature set, and uses the efficient broadcast channels made available in Bluetooth 4.0.

In addition, as a point of comparison between interactive and non-interactive protocols, we collected power traces for a device waking up to handle an incoming connection over Bluetooth 2.1 (without performing any work). This incoming connection consumes an average of 893mJ, which is roughly 65% of the cost of an entire discovery operation. This connection cost scales linearly, which makes interactive protocols impractical for handling many nearby devices.

Reduction in battery life In order to gauge the reduction in battery life of frequently running a discovery and recognition protocol, we collected power traces for various protocol configurations with up to 5 nearby devices over the course of two epochs (30 minutes). For each protocol, we evaluate two different discovery intervals (60 and 120 seconds); existing applications, such as Haggle [8], use a 120 second interval. Since the energy consumption remains the same across two epochs, we extrapolate the energy consumed to a full day (24 hour period), as shown in Table 3.

The Samsung Galaxy Nexus battery has a capacity of 6.48Wh, which we convert to $23,328$J for the purpose of comparisons within the table. With 5 nearby devices, SDDR uses 5.57% of the battery life per day with a 120 second discovery interval; ResolveAddr uses slightly less than SDDR (around 3%), due to the reduced discovery costs. In comparison, the DH+PSI protocol consumes around 150% of the battery over the course of 24 hours. This means that the battery would completely drain within 16 hours, or within only 12.6 hours when using the 60 second discovery interval. IncConn provides a point of reference for the base-line battery life of an interactive protocol—without executing any protocol, it consumes around four times as much energy as SDDR.

As previously mentioned, we assume that each discovery returns 5 *new* nearby devices; in the case of SDDR, this requires computing the shared secret and using the complete set of listen IDs (instead of the matching set) to query the received Bloom filter. In practice, there will not always be 100% churn in nearby devices in each discovery period, meaning that these results are *conservative estimates* of actual energy consumption.

In order to provide a visual comparison between the protocols, we present snapshots of a single 120 second discovery interval for both the SDDR and DH+PSI protocols in Figure 3. These snapshots show the power consumed at each point in time; energy consumption is computed by integrating over a given interval of time. Both protocols initiate a discovery at around the 20 second mark. Since we designed the SDDR protocol to support non-interactive execution, SDDR over BT2.1 can take advantage of executing both the discovery and recognition portions at the same time. Unlike SDDR, the DH+PSI protocol must perform an interactive recognition protocol that takes longer than the discovery process itself, and must be performed individually with each nearby device. In the right half of the plot, both devices handle incoming discovery and recognition requests from nearby devices (5 for SDDR, and 1 for

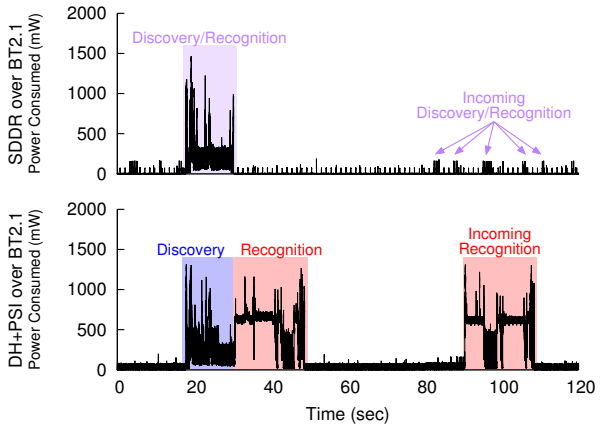

Figure 3: Power traces from running the SDDR and DH+PSI protocols (implemented over Bluetooth 2.1) for one discovery interval of 120 seconds.

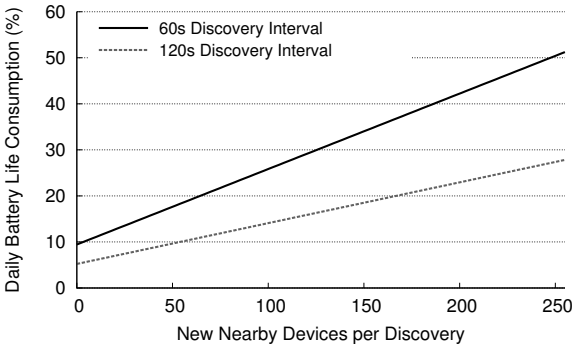

Figure 4: Estimated daily battery life consumption while running the SDDR protocol, for varying numbers of nearby devices.

DH+PSI). Even for the case of a single nearby device, DH+PSI is not practical.

Crowds and DoS attacks A frequently running protocol such as SDDR can potentially open up a new avenue of attack, whereby attackers can try to exhaust the battery of a victim device by forcing it to continually perform new discoveries. Even in benign scenarios, a device may legitimately perform many discoveries over a prolonged interval, e.g., when the user is at a stadium or an indoor auditorium, and the device encounters many other Bluetooth enabled devices. In this section, we experiment with these extreme scenarios, and show that SDDR does not adversely affect battery consumption, regardless of the number of peers it discovers. At the same time, SDDR is able to discover linked peers, and provide reasonable performance even in crowded spaces.

In order to study these worst-case scenarios for SDDR, we estimate the reduction in battery life assuming that there are 255 *new* device responses for every inquiry scan we perform. We use three components of energy consumption for our estimate: idle with Bluetooth running in discoverable mode (E_{BT}), epoch changes (E_{EC}), and discoveries (E_D). Each of these components represents the amount of energy consumed in mJ over the course of a full day (24 hours), and together represent the aggregate energy consumption while running the SDDR protocol:

$$E_{SDDR}(d,i) = E_{BT}(d,i) + E_{EC} + E_D(d,i) \quad (1)$$

$E_{BT}(d,i)$ and $E_D(d,i)$ vary with respect to the number of nearby devices (d), and the discovery interval in seconds (i). We measured the average energy consumption for cases of 1, 3, and 5 discovering devices, for discovery intervals of 60 and 120 seconds. Additionally, we use the results of the microbenchmarks from Table 2 to provide formulas for the energy consumed by epoch changes and discoveries. We assume a linear model, with respect to the number of nearby devices (d), for each i (either 60 or 120 seconds) in computing the formulas for $E_{BT}(d,i)$ and $E_D(d,i)$. The energy consumed by epoch changes (E_{EC}) does not vary with respect to d or i.

We validate Equation 1 by comparing its results to the measured values from Table 3. Without any nearby devices, our estimates are off by 1.96% and 0.66% for 60 and 120 second discovery intervals respectively. Likewise for the case of 5 devices, our estimates deviate from the measurement results by 4.61% and 1.74%.

The estimated daily battery life consumption for varying numbers of nearby devices is shown in Figure 4. Over the course of an entire day, running the SDDR protocol with a 120 second discovery interval consumes 27.82% battery life in this worst-case scenario.

Comparison with SmokeScreen We compare the performance of our protocol with SmokeScreen's discovery protocol [27]. SmokeScreen requires sending one clique signal per advertised ID, and does not use a set-digest data structure (e.g. Bloom filter) to aggregate them. In the authors' implementation, the clique signals are sent over a Bluetooth name request, which holds 248 bytes of data, i.e., roughly 4 clique signals. This makes SmokeScreen less scalable with larger advertised sets: 1) for more than 4 advertised IDs, the clique signals have to be sent over multiple Bluetooth name requests (increasing the discovery latency), and 2) sending multiple name requests for large advertised sets also lead to additional energy consumption. SDDR requires a constant amount of time to detect linkability to a given false positive rate, while SmokeScreen's detection time increases linearly with respect to the number of clique signals.

Since the SmokeScreen measurements were reported several years back, we re-evaluated the energy consumption for SmokeScreen on a recent device. In our measurement, we *conservatively* estimate the energy consump-

tion for SmokeScreen, by measuring only communication related, but not computation- or storage-related energy overhead. Our results suggest that in the case of one nearby device, SmokeScreen's communication consumes $1,628$mJ of energy; for three devices, this increases to $2,071$mJ. Unfortunately, the cost of performing a name request for each individual nearby device is prohibitively expensive. For the same amount of energy spent by SmokeScreen with 3 nearby devices, SDDR can discover and process 35 nearby devices (from Equation 1).

7 Discussion

In this section, we discuss properties and implications of the somewhat unique communication model provided by SDDR, which departs from the norm in two basic ways:

- SDDR decouples confidentiality from identity: encounter peers can communicate securely, even though they do not know each other, and cannot recognize each other during future encounters.
- Communication within SDDR is both defined and limited by radio range, which may not necessarily conform to application semantics.

7.1 Confidentiality without Identity

SDDR's secure encounter primitive provides, in effect, a per-encounter mutual pseudonym for the encounter peers, and an associated shared key. It enables the peers to name each other and communicate securely during their encounter, and at any time after their encounter via an untrusted rendez-vous service. The peers can name and authenticate each other as participants in a specific encounter and communicate securely, while remaining anonymous and unlinkable otherwise (assuming they do not reveal linkable information within their communication). Interestingly, if the users choose, this type of anonymous interaction during an encounter can form the basis for mutual identification and authentication.

Prior systems rely on anonymous or unlinkable encounters between peers, such as SMILE [43] which supports finding missed connections, and SmokeScreen [27] which allows two anonymous peers to exchange addresses for further communication (e.g., E-mail addresses) through the use of a trusted third party service.

7.2 Radio-Range Limited Communication

SDDR communication is limited to radio range, nominally 10 meters for Bluetooth 2.1 (50 meters for the latest Bluetooth 4.0 standard). From an application design point-of-view, range-limited communication may inhibit, but can also prove useful.

Without a third-party data repository or additional protocol mechanisms, e.g., a multi-hop structure, applications that provide notifications among devices beyond radio range cannot be implemented. For example, SDDR cannot be used to replicate the functionality of Google Latitude, which provides updates on friend locations independent of physical distance.

Yet another problem may be that the radio range is not limiting enough! For instance, consider an application that wants to create pairwise encounters and share a group secret only between users who are in the same room. Nothing within SDDR will prevent messages from being received outside the room, enabling a passive eavesdropper to learn the group secret. External mechanisms, in case of the room limited communication, possibly a fast attenuating ultrasound identification beacon are required to manage the impedance mismatch between application semantics and radio range. In general, application designers may choose to use SDDR for a base level of peer detection, and impose criteria that filters unwanted peers.

However, we note that radio range limited communication can have beneficial effects as well. In many situations, the Bluetooth radio range includes the attendees of a socially meaningful event—those with whom a user is likely to interact or share an experience.

Finally, the confidentiality and anonymity provided by SDDR may disproportionately empower abusive users, who could, e.g., spam or otherwise abuse those who are nearby. Here, radio range limited communication provides both a bound on abusive communication and a rudimentary form of accountability. If SDDR is used for malice, the victim is assured that the source of the communication is nearby. The victim could move, or provide evidence of misbehavior (received messages) to law-enforcement authorities. The physical proximity (either of the sender or an accomplice) required for communication within SDDR can potentially serve as a deterrent to abusive communication.

8 Conclusion

In this paper, we articulate the need for efficient secure mobile encounters and their requirements, including selective linkability and efficient revocation. We propose a light-weight protocol called SDDR, which provably meets the security requirements under the random oracle model, and enables highly scalable and energy-efficient implementations using Bluetooth. Experimental results show that our protocol outperforms standard Private Set Intersection by four orders of magnitude. Additionally, its energy efficiency exceeds that of SmokeScreen by an order of magnitude, while supporting stronger guarantees. Energy consumption (and the resulting battery life)

remain practical even under worst-case conditions like dense crowds or DoS attacks.

Acknowledgments

We thank the anonymous reviewers and Jianqing Zhang for their helpful comments, as well as the Max Planck Society. We also thank Rohit Ramesh for his help in integrating SDDR with Haggle. This work was partially supported by the U.S. National Science Foundation under awards IIS-0964541 and CNS-1314857, a Sloan Research Fellowship, as well as by Google Research Awards.

References

[1] AllJoyn. http://www.alljoyn.org.

[2] Bluetooth 2.0 Specification. https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=40560.

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

[4] Federal Information Processing Standards Publication: Digital Signature Standard (DSS). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[5] FireChat. http://www.opengarden.com/firechat.

[6] Foursquare. https://foursquare.com/.

[7] Google Latitude. http://www.google.com/latitude.

[8] Haggle. http://www.haggleproject.org.

[9] LoKast. http://www.lokast.com.

[10] Personal communication with Emiliano De Cristofaro.

[11] Privacy-preserving Sharing of Sensitive Information Toolkit. http://sprout.ics.uci.edu/projects/iarpa-app/code/psst-psi.tar.gz.

[12] StickNFind. https://www.sticknfind.com/.

[13] Tile. http://www.thetileapp.com/.

[14] Venmo. https://venmo.com/.

[15] Whisper. http://whisper.sh/.

[16] Google Fires Engineer for Violating Privacy Policies. http://www.physorg.com/news203744839.html, 2010.

[17] ADITYA, P., ERDÉLYI, V., LENTZ, M., SHI, E., BHATTACHARJEE, B., AND DRUSCHEL, P. EnCore: Private, Context-based Communication for Mobile Social Apps. In *MobiSys* (2014).

[18] ATENIESE, G., DE CRISTOFARO, E., AND TSUDIK, G. (If) Size Matters: Size-hiding Private Set Intersection. In *PKC* (2011).

[19] BAKER, L. B., AND FINKLE, J. Sony PlayStation Suffers Massive Data Breach. http://www.reuters.com/article/2011/04/26/us-sony-stoldendata-idUSTRE73P6WB20110426, 2011.

[20] BLOOM, B. H. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM 13* (1970).

[21] BRESSON, E., CHEVASSUT, O., AND POINTCHEVAL, D. Provably Secure Authenticated Group Diffie-Hellman Key Exchange. *TISSEC 10*, 3 (July 2007).

[22] BRIK, V., BANERJEE, S., GRUTESER, M., AND OH, S. Wireless Device Identification with Radiometric Signatures. In *MobiCom* (2008).

[23] BRODER, A., AND MITZENMACHER, M. Network Applications of Bloom Filters: A Survey. *Internet Mathematics 1* (2004).

[24] CALANDRINO, J. A., KILZER, A., NARAYANAN, A., FELTEN, E. W., AND SHMATIKOV, V. "You Might Also Like: " Privacy Risks of Collaborative Filtering. S&P.

[25] CHAMPION, A. C., ZHANG, B., TENG, J., AND YANG, Z. D-Card: A Distributed Mobile Phone Based System for Relaying Verified Friendships. In *INFOCOM WKSHPS* (2011).

[26] CHANG, Y.-C., AND MITZENMACHER, M. Privacy Preserving Keyword Searches on Remote Encrypted Data. In *Applied Cryptography and Network Security* (2005).

[27] COX, L. P., DALTON, A., AND MARUPADI, V. SmokeScreen: Flexible Privacy Controls for Presence-sharing. In *MobiSys* (2007).

[28] CRISTOFARO, E. D., KIM, J., AND TSUDIK, G. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *ASIACRYPT* (2010).

[29] DIFFIE, W., AND HELLMAN, M. E. New Directions in Cryptography. *IEEE Transactions on Information Theory* (1976).

[30] DONG, C., CHEN, L., AND WEN, Z. When Private Set Intersection Meets Big Data: An Efficient and Scalable Protocol. In *SIGSAC* (2013).

[31] FU, Y., AND WANG, Y. BCE: A Privacy-preserving Common-friend Estimation Method for Distributed Online Social Networks without Cryptography. In *CHINACOM* (2012).

[32] GEHRMANN, C., MITCHELL, C. J., AND NYBERG, K. Manual Authentication for Wireless Devices. *RSA Cryptobytes 7*, 1 (Spring 2004), 29–37.

[33] GOH, E.-J. Secure Indexes. *IACR Cryptology ePrint Archive* (2003).

[34] GOLLAKOTA, S., AHMED, N., ZELDOVICH, N., AND KATABI, D. Secure In-Band Wireless Pairing. In *USENIX Security* (2011).

[35] GREENSTEIN, B., MCCOY, D., PANG, J., KOHNO, T., SESHAN, S., AND WETHERALL, D. Improving Wireless Privacy with an Identifier-free Link Layer Protocol. In *MobiSys* (2008).

[36] GRUTESER, M., AND GRUNWALD, D. Enhancing Location Privacy in Wireless LAN Through Disposable Interface Identifiers: A Quantitative Analysis. In *WMASH* (2003).

[37] HU, Y.-C., AND WANG, H. A Framework for Location Privacy in Wireless Networks. In *SIGCOMM Asia Workshop* (2005).

[38] JARECKI, S., AND LIU, X. Fast Secure Computation of Set Intersection. In *SCN* (2010).

[39] JARECKI, S., AND SAXENA, N. Authenticated Key Agreement with Key Re-use in the Short Authenticated Strings Model. In *SCN* (2010).

[40] KATZ, J., AND YUNG, M. Scalable Protocols for Authenticated Group Key Exchange. *J. Cryptol. 20*, 1 (Jan. 2007).

[41] LAUR, S., AND NYBERG, K. Efficient Mutual Data Authentication Using Manually Authenticated Strings. In *CANS* (2006).

[42] LIN, Y.-H., STUDER, A., HSIAO, H.-C., MCCUNE, J. M., WANG, K.-H., KROHN, M., LIN, P.-L., PERRIG, A., SUN, H.-M., AND YANG, B.-Y. SPATE: Small-group PKI-less Authenticated Trust Establishment. In *MobiSys* (2009).

[43] MANWEILER, J., SCUDELLARI, R., AND COX, L. P. SMILE: Encounter-based Trust for Mobile Social Services. In *CCS* (2009).

[44] MOHAISEN, A., FOO KUNE, D., VASSERMAN, E., KIM, M., AND KIM, Y. Secure Encounter-based Mobile Social Networks: Requirements, Designs, and Tradeoffs. *IEEE TDSC* (2013).

[45] NAGY, M., DE CRISTOFARO, E., DMITRIENKO, A., ASOKAN, N., AND SADEGHI, A.-R. Do I Know You?: Efficient and Privacy-preserving Common Friend-finder Protocols and Applications. In *Proceedings of the 29th Annual Computer Security Applications Conference* (2013).

[46] PORAT, E. An Optimal Bloom Filter Replacement Based on Matrix Solving. *CoRR abs/0804.1845* (2008).

[47] PROSENJIT BOSE AND HUA GUO AND EVANGELOS KRANAKIS AND ANIL MAHESHWARI AND PAT MORIN AND JASON MORRISON AND MICHIEL SMID AND YIHUI TANG. On the false-positive rate of Bloom filters. *Information Processing Letters 108*, 4 (2008), 210 – 213.

[48] RASHID, F. Y. Epsilon Data Breach Highlights Cloud-Computing Security Concerns. http://www.eweek.com/c/a/Security/Epsilon-Data-Breach-Highlights-Cloud-Computing-Security-Concerns-637161/, 2011.

[49] SCHULMAN, A., SCHMID, T., DUTTA, P., AND SPRING, N. Demo: Phone Power Monitoring with BattOr. In *MobiCom* (2011).

[50] SU, J., SCOTT, J., HUI, P., CROWCROFT, J., DE LARA, E., DIOT, C., GOEL, A., LIM, M. H., AND UPTON, E. Haggle: Seamless Networking for Mobile Applications. In *Ubicomp* (2007).

[51] SUN, J., ZHANG, R., AND ZHANG, Y. Privacy-preserving Spatiotemporal Matching. In *INFOCOM* (2013).

[52] THOMAS, K. Microsoft Cloud Data Breach Heralds Things to Come. http://www.pcworld.com/article/214775/microsoft_cloud_data_breach_sign_of_future.html, 2010.

[53] VAUDENAY, S. Secure Communications Over Insecure Channels Based on Short Authenticated Strings. In *CRYPTO* (2005).

[54] WATANABE, C., AND ARAI, Y. Privacy-Preserving Queries for a DAS Model Using Encrypted Bloom Filter. In *Database Systems for Advanced Applications* (2009).

[55] YANG, Z., ZHANG, B., DAI, J., CHAMPION, A. C., XUAN, D., AND LI, D. E-SmallTalker: A Distributed Mobile System for Social Networking in Physical Proximity. In *ICDCS* (2010).

[56] ZHU, F., MUTKA, M., AND NI, L. PrudentExposure: A Private and User-centric Service Discovery Protocol. In *PerCom* (2004).

A Formal Security Definitions and Proofs

A.1 Overview

We follow a standard game-based approach for defining security. We describe a game between an adversary and challenger. The adversary controls the communication medium, and is allowed to schedule the actions of legitimate users. For example, the adversary can instruct a legitimate user to run GenBeacon to generate a discovery beacon; or instruct a recipient to receive the beacon(s) and call Recognize to determine the linkability of discovered neighbors. The adversary can also instruct a legitimate user to perform handshake with any member of the compromised coalition. Link identifiers generated during such a handshake (with the adversary) are marked as compromised (i.e., known to the adversary). In addition, the adversary can explicitly compromise an encounter between two legitimate users in which case the secret link identifier and shared key are exposed; or explicitly compromise a user in which case all its internal states, including previous link identifiers, are exposed.

At some point during the game, the adversary will issue a challenge, either an *anonymity challenge* or a *confidentiality challenge*.

An *anonymity challenge* intuitively captures the notion that an adversary cannot break a legitimate user's anonymity, unless the legitimate user has authorized linkability to a party within the adversary's coalition. Note that this part of the definition captures the unlinkability, selective linkability, and revocability requirements (See Section 3.1) simultaneously.

A *confidentiality challenge* intuitively captures the notion that an eavesdropping cannot learn anything about the (online or post-hoc) communication in between two legitimate users. This is guaranteed since for any two users that remain uncompromised at the end of the security, their shared key established for some time epoch t is as good as "random" to the adversary (assuming their encounter in time epoch t also remains uncompromised by the end of the security game).

A.2 Formal Security Definitions

We define the following security game between and an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$. The time epoch t is initialized to 0 at the beginning of the game. The adversary adaptively makes a sequence of queries as below.

Next time epoch. Increments the current time epoch t.

Expose handshake beacons. The adversary specifies an uncompromised user P_i, identifiers of a subset S_i of P_i's previous encounters, and asks the challenger to expose P_i's handshake beacon in the current time step t using the subset of previous encounters S_i.

Handshake - Uncompromised users. The adversary specifies two uncompromised users P_i and P_j, such that P_j can hear P_i in the current time epoch t. After receiving P_i's handshake beacon, P_j calls the Recognize algorithm, and updates its local state accordingly. The adversary does not obtain information from the challenger for this query.

Handshake - Adversary. The adversary sends a handshake beacon to an uncompromised user P_i. P_i calls the Recognize algorithm, and updates its local state. The identifier of this encounter is marked as compromised. The adversary does not obtain any information from the challenger for this query.

Compromise - Encounter. The adversary specifies a reference to an encounter which took place in time $t' \leq t$ between two uncompromised users P_i and P_j, and the challenger reveals to the adversary the corresponding link identifier, encounter key, and any additional information associated with this encounter.

Compromise - User. The adversary specifies an uncompromised user P_i. The adversary learns all P_i's internal state, including the list of all previous link identifiers, encounter keys, received beacons, and any additional information associated with P_i's previous encounters[3]. P_i and all of its link identifiers are marked as compromised.

Challenge. There can only be one challenge query in the entire game, of one of the following types. In both cases, the adversary outputs a guess b' of b selected by the challenger.

- **Anonymity.** Adversary specifies two users P_i and P_j who must remain uncompromised at the end of the game. The adversary specifies S_i and S_j to the challenger, which (respectively) denote a subset of P_i's and P_j's previous encounters that must remain uncompromised at the end of the game. We require that $|S_i| = |S_j|$. Furthermore, at the end of the game, the adversary must not have issued an "expose handshake beacon" query in the current time step for P_i (or P_j) involving any element in the subset S_i (or S_j).

  The challenger flips a random coin b. If $b = 0$, the challenger constructs P_i's handshake beacon for the current time epoch t for the set S_i, and returns it to adversary. If $b = 1$, the challenger constructs P_j's handshake beacon for the set S_j, and returns it to adversary.
- **Confidentiality.** The adversary specifies two users P_i and P_j who must remain uncompromised at the end of the game.

[3] Specific to our construction, the internal states also include the exponents of P_i's own DH beacons in all previous time epochs.

Furthermore, the encounter between P_i and P_j during time epoch t must also remain uncompromised at the end of the game.

The challenger flips a random coin b. If $b = 0$, challenger returns the encounter key sk_{ij} established between P_i and P_j in time epoch t. If $b = 1$, challenger returns a random number (from an appropriate range).

Definition 1 (Anonymity, Selective linkability)**.** *Suppose that the adversary $\mathscr{A}$ makes a single anonymity challenge in the above security game. The advantage of such an adversary $\mathscr{A}$ is defined as $\mathsf{Adv}^{\mathsf{link}}(\mathscr{A}) := |\Pr[b' = b] - \frac{1}{2}|$. We say that our handshake protocol satisfies selective linkability, if the advantage of any polynomially bounded adversary (making an anonymity challenge) in the above game is a negligible function in the security parameter.*

Definition 2 (Confidentiality)**.** *Suppose that the adversary $\mathscr{A}$ makes a single confidentiality challenge in the above security game. The advantage of such an adversary $\mathscr{A}$ is defined as $\mathsf{Adv}(\mathscr{A})^{\mathsf{conf}} := |\Pr[b' = b] - \frac{1}{2}|$. We say that our handshake protocol satisfies confidentiality, if the advantage of any polynomially bounded adversary (making a confidentiality challenge) in the above game is a negligible function in the security parameter.*

A.3 Proofs of Security

Theorem 1 (Anonymity, selective linkability)**.** *Assume that the CDH problem is hard. For any polynomial-time algorithm $\mathscr{A}$, under the random oracle model,*

$$\mathsf{Adv}^{\mathsf{link}}(\mathscr{A}) \leq \mathsf{negl}(\lambda)$$

where λ is the security parameter.

Proof. If there is an adversary that can break the anonymity game with probability ε, we can construct a simulator which breaks CDH assumption with probability $\frac{\varepsilon}{\mathrm{poly}(N,T,q_o)}$, where N denotes the total number of users, T denotes the total number of epochs, and q_o denotes the number of random oracle queries.

Revealing hashes instead of Bloom filter In the challenge stage, the P_{i^*}'s Bloom filter will have m elements. Instead of announcing the Bloom filter, we assume for the proof that users simply broadcast the outcomes of the hash functions used to construct the Bloom filter. This will only reveal more information to the adversary – so as long as we can prove the security when these hashes are revealed, we immediately guarantee security when the Bloom filter instead of the hash values are revealed.

Real-or-random version and sequence of hybrid games Instead of proving the left-or-right version of the game as in the security definition, we prove the real-or-random version. Namely, the adversary specifies one user P_i (instead of two) in the anonymity challenge (who must remain uncompromised at the end of the game), as well as a subset of P_i's previous encounters (which must remain uncompromised at the end of the game). The challenger flips a random coin, and either returns the faithful hash values to the adversary, or returns a list of random values from an appropriate range. The adversary's job is to distinguish which case it is.

We use a sequence of hybrid games. In the k-th game, replace the k-th hash (out of m hashes) in the challenge stage with some random value from an appropriate range.

Simulator construction The simulator obtains a CDH instance g^{α}, g^{β}. The simulator guesses that the k-th encounter in the anonymity challenge took place between users P_{i^*} and $P_{\widehat{i^*}}$ in time step τ. If the guess turns out to be wrong later, the simulator simply aborts. The simulator answers the following queries:

Expose handshake beacons. First, the simulator generates the DH beacons as below: except for users P_{i^*} and $P_{\widehat{i^*}}$ in time step τ, the simulator generates all other DH beacons normally. For P_{i^*} and $P_{\widehat{i^*}}$ in time step τ, their DH beacons will incorporate g^{α} and g^{β} respectively. Notice that the simulator does not know α, β, or the $\mathsf{dhk} := g^{\alpha\beta}$. Except for $g^{\alpha\beta}$, the simulator can compute all other dhks between two uncompromised users (even when one of g^{α} or g^{β} is involved) since the simulator knows the exponent of at least one DH beacon.

In generating the hashes for the Bloom filter, each hash can correspond to an encounter of the following types:

- Case 1: The hash does not involve an encounter in time τ. The simulator can compute the dhk and link identifier normally in this case.
- Case 2: The hash corresponds to an encounter in time τ, but at least one of the parties in the encounter is an uncompromised user (at the time of the challenge query) other than P_{i^*}, $P_{\widehat{i^*}}$. Notice that the simulator can compute the dhk (and hence the link identifier) in this case, since the simulator knows the exponent of the DH beacon of the other party.
- Case 3: The hash corresponds to an encounter in time τ, and between P_{i^*} and $P_{\widehat{i^*}}$. In this case, the simulator does not know the $\mathsf{dhk} = g^{\alpha\beta}$.
- Case 4: The hash corresponds to an encounter in time τ, and between P_{i^*} (or $P_{\widehat{i^*}}$) and the adversary. Suppose in this encounter in question, the adversary sent P_{i^*} the DH beacon g^{γ}. (The case for $P_{\widehat{i^*}}$ is similar and omitted). The simulator does not know the $\mathsf{dhk} = g^{\alpha\gamma}$ in this case.

Regardless of which type of encounter the hash corresponds to, as long as the simulator knows the dhk of this encounter, it can compute the link identifier and Bloom filters. Below, when we explain how to answer queries of the types "Handshake - Uncompromised users" and "Handshake - Adversary", we will explain how the simulator generates and records a link identifier for each of these encounters – even when it may not know the dhk (Cases 3 and 4). In this way, the simulator can answer queries for Cases 3 and 4 as well.

Handshake - Uncompromised users. Except for the encounter between P_{i^*} and $P_{\widehat{i^*}}$ in time epoch τ, for all other encounters, the simulator can compute the resulting dhks for both uncompromised users – even when one of g^{α} or g^{β} is involved. Therefore, the simulator computes and saves the dhk, which may later be used in answering "expose handshake beacon" queries.

For the encounter between P_{i^*} and $P_{\widehat{i^*}}$ in time τ, since the simulator does not know $\mathsf{dhk} := g^{\alpha\beta}$, it simply chooses a random link identifier and saves it internally, which will later be

used in answer "Expose handshake beacon" queries to construct Bloom filters.

Handshake - Adversary. Except when time step τ and user P_{i^*} or $P_{\widehat{i^*}}$ are involved, the simulator can proceed normally, and generate and dhk and other secrets that are derived as hashes of the dhk.

For time step τ, and P_{i^*} or $P_{\widehat{i^*}}$, something special needs to be done. Assume the adversary sends P_{i^*} handshake beacon g^γ (the case for $P_{\widehat{i^*}}$ is similar and omitted). The simulator does not know α or γ, hence it cannot compute the corresponding $\mathsf{dhk} := g^{\alpha\gamma}$. Without loss of generality, assume $g^\alpha < g^\gamma$. The simulator picks a random link identifier L^* – intended to be the link identifier for this encounter with the adversary. The simulator saves L^*, which will later be used in answering "Expose handshake beacon" queries.

The simulator informs the random hash oracle of the tuple $(\mathsf{L}^*, g^\alpha, g^\gamma)$. Later, random oracle may receive multiple queries of the form $H_0(g^\alpha||g^\gamma||Z)$. Suppose there are at most q_o of these queries. With probability $\frac{1}{q_o+1}$, the hash oracle never uses encK^* as the answer. With probability $1-\frac{1}{q_o+1}$, the hash oracle guesses one of these queries at random, and uses L^* as the answer. The simulator guesses correctly with probability at least $\frac{1}{q_o+1}$ where q_o is the number of hash oracle queries.

Compromise - Encounter. The adversary specifies a reference to a previous encounter (i, j, t'), where users P_i and P_j are uncompromised thus far. If i and j are not i^* or $\widehat{i^*}$, or $t' \neq \tau$, the simulator answers the query normally.

If $t' = \tau$, i and j cannot simultaneously be i^* and $\widehat{i^*}$, otherwise the simulator would have aborted. If one of i or j is i^* or $\widehat{i^*}$, the simulator can still answer the query, even without knowing α or β – since the simulator knows the exponent of the other player's DH beacon.

Compromise - User. If the adversary issues this query for user P_{i^*} or $P_{\widehat{i^*}}$, the simulator simply aborts. For all other uses, the query can be answered normally.

Random oracle. Above, we mentioned how the random oracle handles queries of the form $H_0(g^\alpha||g^\gamma||Z)$, where g^γ was a DH beacon from the adversary in a "Handshake - Adversary" query. For all other random oracle queries, the simulator picks random numbers to answer. The simulator records previous random oracle queries, so in case of a duplicate query, the same answer is given. Whenever the simulator needs to evaluate the hash function, it also queries its own random oracle.

Challenge - Anonymity. The Bloom filter hash values requested in the challenge stage must not have been queried in an "Expose handshake beacon" query. In the k-th hybrid game, the simulator outputs random values for the first k hashes. For the rest, the simulator constructs the answers normally – since these encounters happened before time τ, the simulator can compute their link identifiers and compute these hashes normally.

Without loss of generality, assume that $g^\alpha < g^\beta$. In the above simulation, the simulator makes all guesses correctly with probability at least $\frac{1}{\mathrm{poly}(N,T,q_o)}$. Conditioned on the fact that the simulator made all guesses correctly, unless the adversary queried $H_0(g^\alpha||g^\beta||g^{\alpha\beta})$, the $(k-1)$-th and k-th hybrid games are information theoretically indistinguishable from each other to the adversary. Now the adversary cannot have queried at any point $H_0(g^\alpha||g^\beta||g^{\alpha\beta})$ with more than negligible probability, since otherwise we can construct a simulator that outputs $g^{\alpha\beta}$ with non-negligible probability, thus breaking the CDH assumption.

□

Theorem 2 (Confidentiality)**.** *Assume that the CDH problem is hard. For any PPT algorithm $\mathscr{A}$, under the random oracle model,*

$$\mathsf{Adv}^{\mathsf{conf}}(\mathscr{A}) \leq \mathsf{negl}(\lambda)$$

where λ is the security parameter.

Proof. The simulator guesses that the adversary will issue a confidentiality between users P_{i^*} and $P_{\widehat{i^*}}$ in time epoch τ. If the guess turns out to be wrong later, the simulator simply aborts.

Suppose that simulator gets a CDH instance (g^α, g^β). The simulator would then answer all queries exactly as in the proof of Theorem 1, except for the challenge – instead of submitting a anonymity challenge, the adversary now submits a confidentiality challenge:

Challenge - Confidentiality. If i, j, and current time epoch τ does not agree with what the simulator had guessed, the simulator simply aborts. Otherwise, the simulator would have chosen a random link identifier in a "Handshake - Uncompromised users" query for $(i^*, \widehat{i^*}, \tau)$. The encounter key of this session is obtained by making a random oracle query on $H_2(L)$.

The simulator makes all guesses correctly with probability at least $\frac{1}{\mathrm{poly}(N,T,q_o)}$. Conditioned on the fact that all guesses are correct, the encounter key returned in the challenge stage is information theoretically indistinguishable from random, unless the adversary has queried $H_0(g^\alpha||g^\beta||g^{\alpha\beta})$ (assuming $g^\alpha < g^\beta$ without loss of generality). However, if the adversary makes such a random oracle query with non-negligible probability, we can construct a simulation that leverages the adversary to break the CDH assumption.

□

A.4 Co-Linking

Proposition 1. *Any non-interactive handshake protocol must be subject to a co-linking attack.*

Proof. In an non-interactive protocol, a user Alice publishes a message M in a certain time epoch. Suppose Bob and Charles have met Alice before (in encounters with link-ids L and L' respectively), and Alice has granted both of them permission to link her. Bob should be able to derive from his secret state and the message M, the link identifier L linking this encounter to the previous encounter L. Similarly, with his secret state and the message M, Charles should also be able to derive L'. Now trivially, if Bob and Charles collude, they can decide that the message M can be linked to previous encounters L and L'. □