# Light-Weight Cryptography for Encounter-Based Networking

Paper #*XXX*, 10 pages

## 1. INTRODUCTION

The ubiquitous adoption of intelligent mobile devices with short-range radio and sensing capabilities has fueled the popularity of diverse mobile social applications. In particular, as users come into close range with each other, they form *encounters*. The spatial-temporal graph of encounters encodes rich mobility and social networking information. and enables a wide class of applications, henceforth referred to as *Encounter-based Networking (EbN) applications*. For example, encounter-based networking allows presence sharing with nearby friends, deals from nearby stores; it allows nearby users to communicate with each other and share data; and also allows a user to send *post-hoc* messages to encounter peers, e.g., sending alerts to users who have previously come into contact with a patient infected with a contagious disease.

*Elaine - cite smile, smokescreen, and ebn papers.*

Privacy challenges endure. Most presently deployed mobile context-sensitive solutions rely on a trusted cloud service [2, 3] that is used to match and relay information, requiring users to relinquish control over their sensitive contextual information, perils of which have been extensively noted [4, 7, 13, 6, 11].

On the other hand, solutions that rely on active local radio-to-radio handshakes risk long-term tracking of users' locations if static identifiers are exchanged during the handshake. To protect users from being tracked, ideally ephemeral identifiers should be used instead *Elaine - cite slifi paper*. The naive use of ephemeral identifiers, however, precludes desirable functionality. If users keep evolving their identifier or pseudonym, friends or users with prior trust relations will be unable to recognize each other when they are nearby.

### *Our contributions.*

We propose a *light-weight* cryptographic handshake protocol for EbN that resolves this dichotomy between privacy and functionality, and offers the best of both worlds. While allowing users to use "ephemeral identifiers" to protect their anonymity by default, we enable a form of *selective linkability*, i.e., users with prior trust relations may recognize each other when they are nearby.

From a theoretic perspective, the problem of achieving selective linkability while preserving anonymity by default may be solved using existing primitives such as Private Set Intersection (see Section *Elaine - refer to intuition section*). However, a straightforward adoption of existing primitives does not lead to a power and communication efficient implementation.

In EbN, device discovery is a frequently recurring operation. Therefore, it is imperative that the handshake protocol used for device discovery is extremely light-weight — to minimize power consumption and maximize battery life. To this end, we design a highly efficient handshake protocol, and implemented and evaluated its performance on Bluetooth. Our design and implementation feature the following desirable properties.

- *Non-interactiveness.* Our protocol is non-interactive. Each device periodically (e.g., every 1.5 seconds) broadcasts a single *beacon* message signalling its presence. The beacon message is updated for each time epoch (e.g., every 15 minutes) to ensure long-term unlinkability.

  In each time epoch, this beacon message appears pseudorandom to a neighbor by default; however, if the neighbor has a prior trust relation with the sender of the beacon, the neighbor can recognize the identity of the sender.

- *Power efficient Bluetooth implementation.* The above non-interactive property is crucial to enabling a power-efficient Bluetooth implementation. Our implementation *piggybacks the beacon message over the Bluetooth protocol itself. Thus, once the beacon for a time epoch has been generated, sending a beacon message (e.g., every* 1.5 *seconds) need not wake up the main processor.* We only wake up the main processor periodically (e.g., every 15 seconds) to perform neighbor discovery, and compute the beacon message(s) for the next time epoch(s).

  Our experimental results show that *Elaine - put in* *result on battery life.* We also demonstrate that our protocol is *Elaine - fill in* faster than the naive approach of using a Private Set Intersection protocol *Elaine - forward reference to intuition section*

*Elaine - make a professional comment about this can be implemented on radios other than bluetooth. i dont know how to phrase it.*

- *Formal security modelling and proofs of security.* Last but not the least, we are the first to formally define the security requirements for an EbN handshake protocol (Section 4.4 and Appendix A). We also formally prove the security of our construction (Appendix B).

*Elaine - *******notes below********

[lots of applications make use of short-range encounters between mobile devices.]

[Privacy is important in these applications. if a persistent id is announced, users can be easily tracked. One solution is to announce ephemeral ids for device discovery. However, this naive solution precludes the ability for friends or users with a prior trust relation to recognize each other.]

Our contributions:

– formalize the selectively linkable encounter handshake problem.

– propose a light-weight crypto protocol to achieve the above. the protocol outperforms the naive solution of PSI by several orders of magnitude.

– our protocol is non-interactive, and therefore amenable to power efficient implementation on bluetooth (and potentially other types of radios) we demonstrate that implementation on bluetooth can piggyback on the bluetooth protocol itself, without having to wake up the cpu except for periodically recomputing the cryptographic beacon for the next epoch, and for actively discovering neighbors our protocol is formally proven to be secure.

talk about battery life, and how much additional overhead we have in comparison with normal bluetooth.

## 2. RELATED WORK

*Elaine - 0. trusted cloud based solution; 1. mobile social applications; 2. privacy in these apps*

*Elaine - smile*

*Elaine - smokescreen is closest? it talks about precomputing and power efficiency*

## 3. BACKGROUND AND ASSUMPTIONS

### 3.1 Background on Encounter-based Networking Applications

*Elaine - or maybe instead of background, have a policy/application section at the end? access control policies can be context-aware. discuss these issues and punt to system paper.*

### 3.2 Assumptions

*Elaine - assume MAC layer anonymity, cite slifi*

## 4. OVERVIEW AND DEFINITIONS

### 4.1 Security Requirements

**Long-term unlinkability.** The information exchanged during the handshake reveals no information about the devices participating in the encounter, and cannot be used for recognizing this device or user during a subsequent handshake.

**Selectively linkability.** After a completed handshake between Alice and Bob, Alice may optionally choose to make herself identifiable to Bob in future encounters. During a subsequent handshake with Alice, the protocol executing on Bob's device outputs a link identifier linking Alice to the previous encounter (or previous encounters).

Let $C$ denote a coalition of users, and suppose Alice has *not* granted linkability to any user in the coalition $C$, then all of Alice's encounters are unlinkable to $C$.

*Elaine - note uni-directional here.*

**Revocability.** Alice may, at any time, *unilaterally* revoke Bob's right to recognize her. Consequently, Bob will not be able to recognize Alice for subsequent encounters.

*Elaine - discuss: a user can have several personas.*

### 4.2 Overview and Terminology

*Elaine - explain high level of the protocol, and terminology, such as link identifier, etc. stress that the definition is non-interactive*

*Elaine - fix the following text. right now it sounds more like a solution than prob defn*

In EbN, every encounter generates a unique (except with negligible probability) *link identifier* (denoted $\mathsf{L}$), known only to the two parties involved in the encounter. In each time epoch, every user broadcasts a *handshake beacon* to announce its presence. Intuitively, the handshake beacon encodes the set of link identifiers known to the user. Furthermore, a user only includes an link identifier in the beacon if it allows the remote party in the corresponding encounter to recognize itself.

The encoding must be secure in the following sense: 1) if a user receiving the beacon knows one or more link identifiers in the set, the user can detect their existence; and 2) for any other link identifiers which the user does not know, the user learns nothing about them (except with negligible probability).

Each user stores state about previous encounters it has involved in. For each encounter identified by its link identifier, a user can also additionally store any contextual, application-specific information associated with the link identifier.

## 4.3 Formal Problem Definition

The EbN handshake protocol consists of two algorithms $\mathsf{GenHSBeacon}$ and $\mathsf{PerformHS}$ as described below.

$\mathsf{beacon} \leftarrow \mathsf{GenHSBeacon}(\mathsf{epoch}, \mathsf{link\_ids}, \mathsf{auth\_vec})$: In each time epoch, every user wishing to perform peer discovery executes the $\mathsf{GenHSBeacon}$ algorithm, which takes as input a time $\mathsf{epoch}$, a set of $\mathsf{link\_ids}$ of $m$ previous encounters, and an authorization bit vector $\mathsf{auth\_vec} \in \{0, 1\}^m$, indicating whether the user wishes to let the remote party in each of the $m$ previous encounters recognize itself.

The $\mathsf{GenHSBeacon}$ protocol outputs a beacon message which the user then broadcasts.

$(sk, S, \mathsf{L}_{\text{this}}) \leftarrow \mathsf{PerformHS}(\mathsf{epoch}, \mathsf{beacon}_{\text{remote}}, \mathsf{link\_ids})$: Upon receiving $\mathsf{beacon}_{\text{remote}}$ from a remote party, a user executes the $\mathsf{PerformHS}$ algorithm, which takes in the current time $\mathsf{epoch}$, a $\mathsf{beacon}_{\text{remote}}$ from the remote party, and the set $\mathsf{link\_ids}$ saved for previous encounters.

The $\mathsf{PerformHS}$ protocol then outputs a shared encounter key $sk$ with the remote party (used for encryption and authentication), a set $S \subseteq \mathsf{link\_ids}$ of link identifiers linking the remote party to a set of previous encounters, and the link identifier of this encounter, denoted $\mathsf{L}_{\text{this}}$.

## 4.4 Formal Security Definition

We follow a standard game-based approach for defining security. We describe a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. 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 $\mathsf{GenHSBeacon}$ to generate a handshake beacon; or instruct a recipient to receive the handshake beacon(s) and call $\mathsf{PerformHS}$ 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 a *anonymity challenge* or a *confidentiality challenge*.

*Elaine - make the terminology consistent in appendix as well*

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. In an anonymity challenge, the adversary specifies two uncompromised users $P_i$ and $P_j$, and references to $m$ of each challenge user's previous encounters (which must remain uncompromised by the end of the game). The challenger selects one of $P_i$ or $P_j$ at random, and generates a beacon message involving the $m$ previous previous encounters specified by the adversary. Our security definition (Definition *Elaine - forward reference*) stipulates that no polynomial-time adversary can distinguish whether the challenger selected $P_i$ or $P_j$ in generating the challenge beacon. Note that this part of the definition captures the unlinkability, selective linkability, and revocability requirements (Section 4.1) 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).

The above explains the intuition behind our game-based security definition. The formal security definition is deferred to Appendix *Elaine - forward reference.*

## 5. CRYPTOGRAPHIC HANDSHAKE

### 5.1 Intuition

*Elaine - describe the PSI strawman. think about whether to put it here or in intro*

*Strawman: Diffie-Hellman with PSI.*

Assume two users use a Diffie-Hellman exchange to generate a DH key when they encounter. A secret link identifier can be generated from this DH key, and is known only to the two encounter peers.

Each user $x$ constructs a set $S_x$ of the link identifier of their previous encounters for which they wish to be recognizable in subsequent encounters. Upon encountering each other and performing the DH exchange, users $A$ and $B$ can additionally perform a cryptographic protocol called Private Set Intersection (PSI) [8, **?**, **?**], *Elaine - fill in missing citations* at the end of which they learn the intersection of their sets $S_A$ and $S_B$, without revealing any additional information about other set elements. If the intersection is non-empty, then $A$ and $B$ have chosen to link, and the intersection identifies the previous link identifier. If the intersection is empty, then either $A$ and $B$ have not met before, or they have not granted each other linkability.

For this strawman scheme to be secure, we would additionally require a PSI protocol with *unlinkability* across multiple executions, i.e., when a user executes the PSI protocol multiple times, these multiple executions must not be linkable to each other.

This simple strawman scheme with standard PSI allows us to achieve all desired security properties; however, existing PSI constructions are expensive (See Section **??** *Elaine - fill in reference*), and unsuitable for repeated execution on resource-constrained devices.

*Our approach.*

We wish to achieve the same properties as the PSI strawman, while reducing computation and bandwidth overhead. The key observation is that in our setting, the link identifier are selected uniformly at random, or as cryptographically hard-to-compute secrets (e.g., Diffie-Hellman keys) from a *large* space (sized exponential in the security parameter).

Based on this observation, each party can announce a Bloom filter [9] encoding[1] the set of permitted link identifier. The Bloom filter is constructed with a cryptographic hash (modelled as a random oracle in the proof), which is seeded with a publicly-known *time-dependent nonce*, such that the Bloom filter changes over time and observing a user's filters does not allow an adversary to track the user across time epochs. Further, given a Bloom filter, a remote party gains no information about any link identifier in the filter that they do not already know; brute-force attacks succeed with only negligible probability since the space of link identifier is exponential in size in terms of the security parameter.

*Unidirectional* linkability may be achieved, since including a link identifier in the announced Bloom Filter would unidirectionally allow the other party to recognize oneself in a future encounter. *Revocability* may be achieved, since a link identifier may be removed from the announced Bloom Filter at any time.

*Elaine - explain why advertiseID and listenID, to avoid attack*

*Splitting a link identifier into* listenID *and* advertiseID*.*

One subtle issue is that for the protocol to be secure, two encounter peers cannot use the same link identifier in their announced Bloom Filter, since otherwise, an eavesdropper may be able to observe that the two parties have encountered before. For example, suppose Alice and Bob encounter, and generate a common link identifier $L$. Next, Alice and Bob both announce a Bloom Filter containing only $L$. In this case, their Bloom Filters will be identical, and an eavesdropper can easily determine that Alice and Bob have encountered before.

[1]Technically, any set digest structure is sufficient. We have implemented our protocol using Matrix filters [10] as well; we use Bloom filters because of a very efficient encoding possible over Bluetooth.

**Inputs:** Each participant $P_i$ has a listen set $LS_i$, and an advertised set $AS_i$.
*; To grant linkability, include a corresponding link identifier in $A_i$.*

**Outputs:** Every user $P_i$ outputs:
1. $\forall$ user $P_j$ that $P_i$ discovers ($j \neq i$):
2. A shared key $sk_{ij}$
3. A potential $\mathsf{listenID}_j \in LS_i$ linking $P_j$ to a previous encounter
4. A link identifier pair $(\mathsf{advertiseID}_j,\ \mathsf{listenID}_j)$

**Protocol:** Each $P_i$ performs the following steps:

<u>GenHSBeacon :</u>

1. Select random $a_i \in_R \mathbb{Z}_p$
   *; Ephemeral, lasts for one epoch.*
2. Broadcast $(g^{a_i}, \mathsf{BF}_i)$, where
   $\mathsf{BF}_i := \mathsf{BF}\{H(1||g^{a_i}||a) : a \in AS_i\}$

<u>PerformHS :</u>

3. For each user $P_j$ ($j \neq i$) that $P_i$ can hear, compute:

- DH value $\mathsf{DHKey}_{ij} = (g^{a_j})^{a_i}$
- Link $\mathsf{L}_{ij} := \begin{cases} H(0||g^{a_i}||g^{a_j}||\mathsf{DHKey}_{ij}) \text{ if } g^{a_i} < g^{a_j} \\ H(0||g^{a_j}||g^{a_i}||\mathsf{DHKey}_{ij}) \text{ otherwise} \end{cases}$
- Encounter key $sk_{ij} := H(3||\mathsf{L}_{ij})$
- Matching set $\mathsf{M}_j := \{\ell : \ell \in LS_i \wedge H(1||g^{a_j}||\ell) \in \mathsf{BF}_j\}$
- If $g^{a_i} < g^{a_j}$: $\mathsf{listenID}_j \leftarrow \mathsf{L}_{ij}$, $\mathsf{advertiseID}_j \leftarrow \mathsf{L}_{ij}$ xor 1
  Else : $\mathsf{advertiseID}_j \leftarrow \mathsf{L}_{ij}$, $\mathsf{listenID}_j \leftarrow \mathsf{L}_{ij}$ xor 1
- Link identifier pair $(\mathsf{advertiseID}_j, \mathsf{listenID}_j)$.

Figure 1: EbN non-interactive handshake protocol. For notational reasons, we present the group operations using multiplicative notation. *Elaine - this seems a bit confusing. to fix. the number 3 in the hash is not contiguous*

For this reason, we split a link identifier into a listenID and an advertiseID. Specifically, one encounter peer will announce advertiseID $:= L$ and listen for listenID $:= L$ xor 1; the other party will listen for listenID $:= L$ and announce advertiseID $:= L$ xor 1. We henceforth refer to (listenID, advertiseID) as a *link identifier pair*.

## 5.2 Detailed Handshake Protocol

*Elaine - the terminology in the protocol figure differs from problem definition, needs explanation*

To start with, a participant $P_i$ has a set of listenIDs, called a *listen set*; and a set of advertiseIDs, called an *advertise set*. For a previous encounter identified by the link identifier pair (listenID, advertiseID), if $P_i$ wishes the encounter peer to recognize itself, it includes advertiseID in its *advertise set* — which will be incorporated in to the Bloom Filter beacon to be announced. The rest of the handshake protocol proceeds as in Figure 1.

## 5.3 Extensions and Limitations

*Hiding Bloom filter load.*

By aggregating multiple Bloom filters, and calculating the distribution of the number of bits set, it is possible to determine the number of included entries (size of the user's advertised set). This leaks information from the execution of the protocol that could be used to link encounters. Therefore, we declare a global maximum number of elements $N$. When the size of the advertised set is less than $N$, we insert additional random elements as padding. Rather than computing the hashes, we simply select $K * (N - |\mathsf{advertiseIDs}|)$ bits to set to 1, where $K$ is the number of hash functions.

*Explicit verification to eliminate BF false positives.*
Our use of Bloom Filters can lead to false positives, i.e., falsely identify a peer as a previously encountered peer. We can eliminate such false positives by introducing an extra explicit verification message at the end of the handshake. When two parties $A$ and $B$ have detected a non-empty intersection, they perform a verification step to verify that the other party really has the same secret link-id pair. To prove to a remote party that one knows a secret link identifier $L$ ($L$ can be either a listen ID or an advertised ID), one simply sends the other the verification message $\langle \text{nonce}, H(4||L||\text{nonce})\rangle$. *Elaine - renumber the number 4 in the hash...*

*Defenses against man-in-the-middle.*
Absent any pre-established secrets or an out-of-band channel, man-in-the-middle attacks are unavoidable, e.g., as in the Diffie-Hellman key exchange.

In EbN, it suffices to secure the first handshake between two peers against man-in-the-middle — once the two peers securely establish a shared secret in the first encounter, future encounters are no longer susceptible to a man-in-the-middle attack.

For two peers with trust relations, we can easily defend against man-in-the-middle attacks by relying on a secret established prior to the first encounter between these two peers. One simple way to do this is *Elaine - describe it here.*

An alternative way to secure against man-in-the-middle is to rely on out-of-band channel. For example, one way is to display on the two users' screens a visual barcode that is dependent on the shared key. The two users then check to ensure that they have the same bar code. This approach has been well-studied in prior work. *Elaine - cite mana papers.*

*The co-linking attack.*
Our handshake protocol is open to a potential *co-linking* attack — but we show that any non-interactive protocol must be prone to this limitation.

In our protocol, Alice is unlinkable to a coalition of users if Alice has not granted linkability to anyone in that coalition. However, if Alice encountered Bob and Charles in two separate encounters (whose link-ids are $L$ and $L'$), and has granted both of them permission to recognize her in the future, Bob and Charles can potentially perform a co-linking attack: by combining their secret states $L$ and $L'$, in a future handshake with Alice, they can jointly discover that encounters $L$ and $L'$ correspond to the same person.

In Appendix B.1, we show that any non-interactive protocol must be subject to the co-linking attack. Given the relatively harmless nature of the "attack" (two of Alice's friends can recognize that they have a mutual friend ), we have chosen a non-interactive protocol for EbN due to the power efficiency gains.

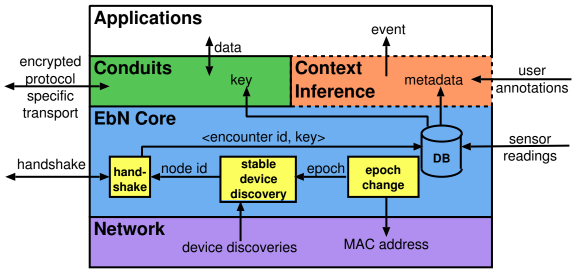

Figure 2: EbN architecture. *Elaine - Matt: cite ebn system paper here. i tried but compiler complains*

## 6. IMPLEMENTATION

### 6.1 Architecture

We have implemented EbN as a network level service that offers a simple abstraction and API to application developers (Figure 2). While not the focus of this paper, for completeness, we give a high-level overview of the EbN architecture below. A detailed exposition of the EbN architecture, is provided in EbN system design paper [5].

*Elaine - just draw a figure and just cite system paper. will not claim this as a contribution, just to give some background. should not be very long.*

*Elaine - Matt, can you please write this.*

### 6.2 Android Bluetooth 4.0 Implementation

We have implemented EbN on the Android platform. The codebase is written in C++ and Java, runs with root privileges, and consists of 6800 lines of code. The implementation uses Bluetooth 4.0 (BT4) to detect and communicate with neighboring devices. For development and evaluation, we use Samsung Galaxy Nexus phones running Android 4.1.2 with the android-omap-tuna-3.0-jb-mr0 kernel. We modified the kernel by fixing a power-related bug in the Bluetooth controller software[2].

[2]The software used a 1s timer, which on timeout would set the hardware to enter low-power mode (LPM). Activity be-

We utilize several of the device roles added in BT4, including *advertisers* and *scanners*. Advertisers periodically broadcast small ($\leq$ 31 bytes) messages, while scanners passively listen for these advertisements. Advertisers and scanners can optionally participate in *active scanning*: advertisers note their willingness to provide an additional payload upon request, while scanners submit requests for this data after receiving the advertisement [1].

*Handshake Protocol.*

During each epoch, we need to broadcast our 192-bit ECDH public key and Bloom filter. Including the full public key in each advertisement would leave $<$ 7 bytes for the Bloom filter, resulting in slow set intersection convergence rates. Therefore, we need a way to divide the public key amongst multiple advertisements. This division should allow devices to form an encounter after receiving a fixed number of advertisements, and afterwards continue to improve their linkability confidence through the Bloom filters.

We note that the advertisement channel can be considered a packet erasure channel, i.e. packets are either received correctly or not at all. Therefore, we use Reed-Solomon (RS) coding [12] for the DH public key. RS codes are optimal erasure-correction codes, meaning the receiver can reconstruct the original $K$ data symbols from any $K$ data and redundancy symbols. We do not need to perform a similar coding for the Bloom filter, as we can set any segments we do not receive to all ones.

*Epoch change.*

We perform the periodic epoch changes using the private (random) address support in BT4, which does not require a hardware reset. We generate the new address using the algorithm described in Section **??**; however, we include the compressed y-coordinate (1 bit) of the ECDH public key as part of the random nonce.

*Broadcast Beacons - Advertisements.*

Each advertisement contains the next advertisement number $a$, the $a^{\text{th}}$ RS coded symbol of the public key, and the $a^{\text{th}}$ segment of the Bloom filter. We update the advertisement data at the start of each discovery period, which the Bluetooth controller periodically broadcasts according to the specified interval.

*Discovery.*

We scan for advertisements, using a duration slightly longer than the advertisement interval, in order to capture broadcasts from nearby devices. For each device, we aggregate advertisements until we are able to decode the public key for the epoch (requires $K$ advertisements). Once decoded, we use the public key to query the Bloom filter for all link values in the matching set (initially equal to listenIDs). At the end of the discovery period, we notify the EbN Controller of any new shared keys or matching set updates.

tween the host and controller resets the timer, which can occur right before a system suspend (an operation that takes less than 1s). We changed this timer to 100ms, allowing the hardware to correctly enter LPM prior to the system suspending.

## 7. EXPERIMENTAL RESULTS

### 7.1 Comparison with Private Set Intersection

We measure the EbN handshake protocol computation time while varying the number of linkable identifiers, and compare the elapsed time to that reqired for a PSI protocol. We use an implementation [**?**] of the JL10 scheme [**?**], one of the fastest schemes known-to-date that is secure under the malicious model, and can be nodified to achieve unlinkability across multiple sessions. Both protocols are executed using a single core on the Samsung Galaxy Nexus platform (1.2 GHz ARM Cortex-A9 processor).

Figure 3 shows the runtimes for each protocol; each bar is an average of 50 runs, with error bars denoting the 5th and 95th percentile values. We divide EbN into two separate trials, each processing a different number of advertisements in order to achieve the specified false positive rates. We note that additional advertisements do not add much computation time in EbN, since they only query the Bloom filter for remaining entries in the matching set. Even though PSI always provides an exact answer, EbN allows for far better efficiency (over two orders of magnitude faster) by slightly relaxing that requirement. The gain in performance is crucial for practical deployment as these computations occur during each handshake. PSI protocols do not hide the number of entries in the set, and (if deployed) would require a relative large (128-256) set size. *Matt - We should check out a paper called 'Size Matters: Size-Hiding Private Set Intersection'.*

### 7.2 Power Consumption

*Elaine - Matt: can you please put in power consumption results. the microbenchmark table.*

*Elaine - Matt: can you add a figure showing under typical number of peers, the battery life of the device, and how much we have shortened it in comparison with 1) not running any bluetooth. 2) running bluetooth handshake but not ebn; and 3) running ebn.*

*Elaine - Matt: for the microbenchmark, can we also compare ebn's handshake with the bluetooth handshake, and see what the additional overhead is. we can discuss how to do this if you want.*

## 8. CONCLUSION

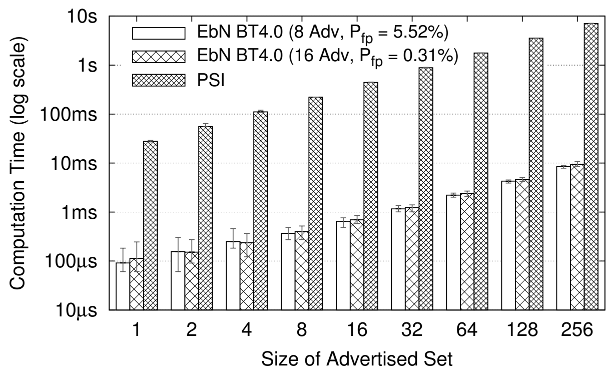

Figure 3: Protocol execution times of PSI versus EbN for an encounter with varying sizes of advertised sets of link values

## 9. REFERENCES

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

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

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

[4] Google fires engineer for violating privacy policies. http://www.physorg.com/news203744839.html, 2010.

[5] Ebn: A communication system based on encounters. Manuscript under submission, 2012.

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

[7] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov. "You might also like:" Privacy risks of collaborative filtering. In *S&P*, 2011.

[8] S. Jarecki and N. Saxena. Authenticated key agreement with key re-use in the short authenticated strings model. In *SCN*, 2010.

[9] D. E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching.* Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998. ISBN 0-201-89685-0.

[10] E. Porat. An optimal bloom filter replacement based on matrix solving. *CoRR*, abs/0804.1845, 2008.

[11] 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/, 2011.

[12] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial & Applied Mathematics*, 8(2):300–304, 1960.

[13] K. Thomas. Microsoft cloud data breach heralds things to come. http://www.pcworld.com/article/214775/microsoft_cloud_data_breach_sign_of_future.html, 2010.

## APPENDIX

## A. SECURITY DEFINITIONS

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

- **Next time epoch.** Current time epoch $t$ is incremented.
- **Expose handshake beacons.** The adversary specifies an uncompromised participant $P_i$, 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 between two uncompromised participants $(i, j)$.** The adversary specifies two uncompromised participants $P_i$ and $P_j$. This means that $P_j$ can hear $P_i$ in the current time epoch $t$. After receiving $P_i$'s handshake beacon, $P_j$ calls the PerformHS algorithm, and updates its local state accordingly. The adversary does not obtain information from the challenger for this query.
- **Handshake with the adversary.** The adversary specifies an uncompromised participant $i$, and sends a handshake beacon to $P_i$. $P_i$ calls the PerformHS 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 $(i, j, t')$.** The adversary specifies a reference to an encounter previously taken place in time $t' \leq t$ between two uncompromised participants $P_i$ and $P_j$, and the challenger reveals to the adversary the corresponding link identifier, the encounter key, and any additional information associated with this encounter.
- **Compromise user $i$.** The adversary specifies an uncompromised user $P_i$. The adversary learns all $P_i$'s internal states, including the list of all previous link identifiers, encounter keys, beacons received from remote parties, and any additional information associated with $P_i$'s previous encounters[3].

[3]Specific to our construction, the internal states also include

$P_i$ 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:
  - **Linkability challenge.** Adversary specifies two participants $P_i$ and $P_j$ who must remain uncompromised at the end of the game. Let $S_i$ denote a subset of participant $P_i$'s previous encounters which must remain uncompromised at the end of the game. Let $S_j$ denote a subset of participant $P_j$'s previous encounters which must remain uncompromised at the end of the game. The adversary specifies $S_i$ and $S_j$ to the challenger. 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 participants $i$ (or $j$) 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 challenge.** The adversary specifies two participants $P_i$ and $P_j$ who must remain uncompromised at the end of the game. Furthermore, the encounter between $P_i$ and $P_j$ during time epoch $t$ must also remain uncompromised at the end of the game. 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).

  Finally, the adversary outputs a guess $b'$ of $b$.

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

Definition 2 (Confidentiality). *Suppose that the adversary $\mathcal{A}$ makes a single confidentiality challenge in the above security game. The advantage of such an adversary $\mathcal{A}$ is defined as $\mathsf{Adv}(\mathcal{A})^{\mathsf{conf}} := |\Pr[b' = b] - \frac{1}{2}|$. 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.*

the exponents of its own DH beacons in all previous time epochs.

## B. PROOFS OF SECURITY

Theorem 1 (Selective linkability). *Assume that the Computational Diffie-Hellman (CDH) problem is hard. For any polynomial-time algorithm $\mathcal{A}$, under the random oracle model,*

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

*where $\lambda$ is the security parameter.*

Proof. If there is an adversary that can break the selective linkability game with probability $\epsilon$, we can construct a simulator which breaks CDH assumption with probability $\frac{\epsilon}{\text{poly}(N,T,q_o)}$, where $N$ denotes the total number of participants, $T$ 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 participants 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 participant $P_i$ (instead of two) in the linkability 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 construction is as follows. The simulator obtains a CDH instance $g^\alpha, g^\beta$.

The simulator guesses that the $k$-th encounter in the linkability challenge took place between participants $i^*$

and $\widehat{i^*}$ in time step $\tau$. If the guess turns out to be wrong later, the simulator simply aborts.

The simulator answers queries as below:

- **Expose handshake beacons.** First, the simulator generates the DH beacons as below: except for participants $i^*$ and $\widehat{i^*}$ in time step $\tau$, the simulator generates all other DH beacons normally. For participants $i^*$ and $\widehat{i^*}$ in time step $\tau$, their DH beacons will incorporate $g^\alpha$ and $g^\beta$ respectively. Notice that the simulator does not know $\alpha$, $\beta$, or the $\mathsf{DHKey} := g^{\alpha\beta}$.

  Except for $g^{\alpha\beta}$, the simulator can compute all other $\mathsf{DHKey}$s between two uncompromised participants, even when one of $g^\alpha$ or $g^\beta$ is involved – since the simulator knows the exponent of at least one DH beacon.

  Next, to generate the bloom filters hashes. Each hash can correspond to an encounter of the following types:

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

  Regardless of which type of encounter the hash corresponds to, as long as the simulator knows the $\mathsf{DHKey}$ of this encounter, it can compute the link identifier and bloom filters. Below, when we explain how to answer queries of the types "handshake between two uncompromised participants" and "handshake with an 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 $\mathsf{DHKey}$ (Cases 3 and 4). In this way, the simulator can answer queries for Cases 3 and 4 as well.

- **Handshake between two uncompromised participants.** Except for the encounter between $i^*$ and $\widehat{i^*}$ in time epoch $\tau$, for all other encounters, the simulator can compute the resulting $\mathsf{DHKey}$s for both uncompromised participants – even when one of $g^\alpha$ or $g^\beta$ is involved. Therefore, the simulator computes and saves the $\mathsf{DHKey}$, which may later be used in answering "expose handshake beacon" queries.

  For the encounter between $i^*$ and $\widehat{i^*}$ in time $\tau$, since the simulator does not know $\mathsf{DHKey} := g^{\alpha\beta}$, 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 with the adversary.** Except when time step $\tau$ and participants $i^*$ or $\widehat{i^*}$ are involved, the simulator can proceed normally, and generate and $\mathsf{DHKey}$ and other secrets that are derived as hashes of the $\mathsf{DHKey}$.

  For time step $\tau$ and participants $i^*$ or $\widehat{i^*}$. Something special needs to be done. Assume the adversary sends $i^*$ handshake beacon $g^\gamma$. (The case for $\widehat{i^*}$ is similar and omitted.) The simulator does not know $\alpha$ or $\gamma$, hence it cannot compute the corresponding $\mathsf{DHKey} := g^{\alpha\gamma}$. Without loss of generality, assume $g^\alpha < g^\gamma$. The simulator picks a random link identifier $\mathsf{L}^*$ – intended to be the link identifier for this encounter with the adversary. The simulator saves $\mathsf{L}^*$, 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 $\mathsf{encK}^*$ as the answer. With probability $1 - \frac{1}{q_o+1}$, the hash oracle guesses one of these queries at random, and uses $\mathsf{L}^*$ 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** $(i, j, t')$. The adversary specifies a reference to a previous encounter $(i, j, t')$, where participants $P_i$ and $P_j$ are uncompromised thus far. If $i$ and $j$ are not $i$ or $i^*$, 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 $\alpha$ or $\beta$ – since the simulator knows the exponent of the other player's DH beacon.

- **Compromise user.** If the adversary issues a "compromise user" query for user $i^*$ or $\widehat{i^*}$, the simulator simply aborts. For all other uses, the "compromise user" 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 $g^{\gamma}$ was a DH beacon from the adversary in a "handshake with the 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.
- **Linkability challenge.** The BF hash values requested in the challenge stage must not have been queried in an "expose handshake beacon" query.

  In the $k$-th hybrid game: for the first $k$-th hashes, the simulator outputs random values. For the rest, the simulator constructs the answers normally – since these encounters happened before time $\tau$, 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 $\frac{1}{\text{poly}(N,T,q_o)}$. 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 Computational Diffie-Hellman assumption.

□

THEOREM 2 (CONFIDENTIALITY). *Assume that the Computational Diffie-Hellman (CDH) problem is hard. For any PPT algorithm $\mathcal{A}$, under the random oracle model,*

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

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

Suppose that simulator gets a CDH instance $(g^{\alpha}, g^{\beta})$. The simulator would then answer all queries exactly as in the proof of Theorem 1, except for the challenge – instead of submitting a linkability challenge now, the adversary now submits a confidentiality challenge:

- **Confidentiality challenge** $(i, j)$**.** If $i$, $j$, and current time epoch $t$ 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 between two uncompromised participants" query for $(i^*, \widehat{i^*}, \tau)$. The encounter key of this session is obtained by making a random oracle query on $3||L$.

The simulator makes all guesses correctly with probability at least $\frac{1}{\text{poly}(N,T,q_o)}$. 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. □

## B.1 Co-Linking Attack

PROPOSITION 1. *Any non-interactive handshake protocol must be subject to the 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 $L'$. □