Techu: Open and Privacy-Preserving Crowdsourced GPS for the Masses

Ioannis Agadakos
Dept. of Computer Science
Stevens Institute
of Technology
iagadako@stevens.edu

Jason Polakis
Dept. of Computer Science
University of Illinois
at Chicago
polakis@uic.edu

Georgios Portokalidis
Dept. of Computer Science
Stevens Institute
of Technology
gportoka@stevens.edu

ABSTRACT

The proliferation of mobile devices, equipped with numerous sensors and Internet connectivity, has laid the foundation for the emergence of a diverse set of crowdsourcing services. By leveraging the multitude, geographical dispersion, and technical abilities of smartphones, these services tackle challenging tasks by harnessing the *power of the crowd*. One such service, Crowd GPS, has gained traction in the industry and research community alike, materializing as a class of systems that track lost objects or individuals (e.g., children or elders). While these systems can have significant impact, they suffer from major privacy threats.

In this paper, we highlight the inherent risks to users from the centralized designs adopted by such services and demonstrate how adversaries can trivially misuse one of the most popular crowd GPS services to track their users. As an alternative, we present Techu, a privacy-preserving crowd GPS service for tracking Bluetooth tags. Our architecture follows a hybrid decentralized approach, where an *untrusted* server acts as a bulletin board that collects reports of tags observed by the crowd, while observers store the location information locally and only disclose it upon proof of ownership of the tag. Techu does not require user authentication, allowing users to remain anonymous. Our security analysis highlights the privacy offered by Techu, and details how our design prevents adversaries from tracking or identifying users. Finally, our experimental evaluation demonstrates that Techu has negligible impact on power consumption, and achieves superior effectiveness to previously proposed systems while offering stronger privacy guarantees.

Keywords

Crowd GPS, BLE Tags, Location-based Services, User Tracking, Location Privacy, Privacy-Preserving Protocol

CCS Concepts

•Security and privacy → Domain-specific security and privacy architectures; *Embedded systems security;* **•Networks → Location based services;**

MobiSys '17, June 19–23, 2017, Niagara Falls, NY, USA.

DOI: http://dx.doi.org/10.1145/3081333.3081345

1. INTRODUCTION

The widespread adoption of smartphone devices has transformed many aspects of everyday life, and led to the emergence of a plethora of novel services that range from the menial to critical. This availability of a massive number of geographically dispersed mobile devices with internet connectivity has lent significant traction to the crowdsourcing paradigm and numerous incarnations have surfaced, from collecting real-time traffic data [56] to providing early warning for earthquakes [26].

Crowd GPS is one such type of service that utilizes a network of users, *a crowd*, to locate lost or misplaced objects, or monitor the location of pets [8] and "vulnerable" individuals like children [46] and elders [16]. It has taken off after the release of Bluetooth Low Energy (BLE), which allows near-range communication with a significantly reduced power consumption [31], as it facilitated the creation of small, battery-powered, BLE-enabled tags that can be easily attached to objects, or worn by people and pets. The loss or misplacement of an item is a problem that has tormented many. In 2013 alone, 26,000 articles were forgotten in New York's transportation system [13] and a survey [3] found that one in five people lose or misplace an item every week. This has lead to the rapid growth of crowd GPS, with services such as TrackR boasting over 5 million users.

Despite the obvious benefits of crowd GPS, the designs of existing systems suffer from various drawbacks. Most notably, the location of personal items, and consequently of the user, is collected centrally by the crowd GPS service, which is frequently run by the vendors offering BLE tags for sale. This poses a significant privacy threat, as a large body of research on location privacy in other applications has also highlighted [12, 36, 58, 65]. Second, the ecosystem is fragmented, forming disjoint crowds of users based on the vendor they have purchased tags from. Crowd sourcing performance improves as the size of the crowd grows, hence, this fragmentation limits the effectiveness of current systems.

In this work, we first audit TrackR, one of the most popular crowd GPS services, and expose a vulnerability that enables the misuse of the service for tracking the location of *any* tag owned by a customer. Our findings highlight a critical risk of crowd GPS; *current centralized designs expose a single-point of failure, where vulnerabilities can result in the exfiltration of the location of users.*

Previous works on locating lost or stolen Bluetooth-enabled tags suffer several shortcomings. First, and foremost, they present privacy threats to participating users; in [22, 70] users can be trivially tracked by any member of the crowd by supplying the tag's identifier, while in [70] the service also possesses coarse-grained location data that can lead to deanonymization [24]. Second, [28, 70] follow a "search after the fact" approach, where the crowd searches

for a tag *after* the query is issued, i.e., participants do not maintain information of recently seen tags; this can severely limit the effectiveness of the system, especially in areas with few users.

To ameliorate the aforementioned drawbacks, we propose Techu[1], an open crowd GPS system designed to overcome the above issues and ensure the privacy of users participating in the crowd. In Techu we follow a hybrid decentralized architecture, where reports about tag observation events are kept in an untrusted public server, while the actual location information is stored in a distributed manner. In a nutshell, Techu works as follows. Users participating in our system download a client side app and pair their smartphones with their BLE tags they want to keep track of. The app periodically creates a randomly generated secret for each paired tag, and stores it on the tag as the `DeviceName`, which is advertised by the tag. Participants log the `DeviceName` of all the tags in range that belong to other users, and create observation reports that contains an identifier derived by the `DeviceName`, a timestamp, and their unlinkable communication pseudonym (`ObserverID`). Reports are periodically uploaded in batches to a centralized *untrusted* database that serves as a bulletin board. If a tag is lost, the owner initiates the object-discovery process, and uses the last generated `DeviceName` to retrieve the most recent observation reports for that tag from the server. The owner then uses the `ObserverID` to communicate with the observer over an ephemeral communication channel, through the Google Cloud Messaging (GCM) infrastructure [33]; after proving ownership of the tag, the observer discloses the location at which the tag was seen. By maintaining reports of recently observed tags, Techu allows for the tracking of tags even in sparsely populated areas; it is sufficient for a single user to have been in range of the tag at some point in time, for the owner to obtain information about the tag's location.

We conduct an extensive experimental evaluation of our system, and quantify performance in terms of storage requirements, power consumption, communication costs, and query overhead. As the prevailing role for participants is that of the observer, the overhead introduced by our protocol is of key importance. Apart from the baseline cost of BLE scanning, which is shared by all crowd GPS systems, under "typical" mobility patterns our system requires less than 30 seconds of client-side computation per day (for all the required calculations plus the time required to report to the Techu service) and client storage requirements are less than 55KB. Storage requirements for the bulletin board are very moderate, reaching approximately 23 TB for 12 million participating users with typical mobility patterns. Our system's impact on the battery lifetime of tags is negligible, with less than a 2% reduction. Finally, the cost for owners to locate lost tags is very low, as the relevant observation records that have to be fetched are $\sim$2MB and the communication protocol with an observer is completed within a few seconds. Overall, our system introduces minimal overhead while offering significant privacy properties compared to existing services or proposed solutions.

The key contributions of our work are the following:

- We conduct a security analysis of TrackR, a prominent crowd GPS service, and discover a vulnerability that enables adversaries to misuse the service, effectively turning the participants into a *malnet* [44] that can track the location of any user through time.
- We present Techu, a crowdsourced GPS system designed to effectively locate BLE tags, while guaranteeing the location privacy of participants. Our system is built around an *untrusted* server that can be adopted by vendors or used as a public bulletin board, and it does not require user enrollment or authentication, allowing users to anonymously participate in the crowd.
- We present a series of extensions to our basic design, and analyze the tradeoff between the security guarantees of our system and performance overhead.
- We experimentally evaluate Techu, and find that our client-side and server-side requirements and overhead are minimal, while our protocol allows users to efficiently obtain the required information while preserving their privacy. Techu achieves its goals while reducing the operating costs and without the implications of proprietary "security through obscurity" approaches, enabling the deployment of our system either as community effort or by a tag-selling vendor.

[1]Named after the Egyptian god of record-keeping.

The remainder of the paper is structured as follows. Section 2 provides the necessary background information and privacy risks of existing crowd GPS services, while Section 3 presents our security analysis of a prominent service. In Section 4 we describe our system design and privacy extensions, while Sections 5 and 6 outline the security guarantees and implementation details of our system, respectively. We continue with the experimental evaluation of Techu in Section 7, and discuss limitations and future work in Section 8. Finally, we present related work in Section 9 and conclude in Section 10.

2. BACKGROUND

Here we provide background information on crowd GPS and discuss the privacy risks of its current instantiations.

2.1 Crowd GPS

In crowd GPS the location of objects is established through crowd sourcing, i.e., by enlisting the services of a number of individuals that form a network, referred to as a *crowd*, typically through the Internet. This model is appropriate for tracking objects and devices that do not, or cannot, feature a GPS sensor and Internet connectivity to report their location to their owners. Instead, the objects can be sensed, typically through wireless sensors, by the individuals participating in the crowd, who can establish their location and, hence, help locate the objects.

This is an instantiation of *crowd sensing*, where user devices become part of a network collecting locally observable information through their sensors and uploading it to the cloud, to be aggregated and analyzed at a large scale. This is usually achieved through dedicated client-side apps running on devices carried by users, such as smartphones. Crowd sensing may require different levels of user participation [51], ranging from active participation (*participatory crowd sensing*) [40] to a completely transparent process for end users (*opportunistic crowd sensing*) [55]. A detailed taxonomy of crowd sensing can be found in [38].

Crowd GPS has gained a lot of traction since the introduction and proliferation of BLE, a low-energy Bluetooth network technology. There are multiple vendors (e.g., Tile [41] and TrackR [42]) offering BLE tags, or beacons, that can be attached on objects (e.g., keys and backpacks) or come in the form of bracelets that can be worn by vulnerable individuals [16, 46]. Users that purchase such tags become part of a vendor-maintained crowd by installing an app on their smartphone. The app automatically reports the location of sensed tags and enables users to query the vendor to locate the tags. Most apps, like TrackR and Tile, only track and report their own tags and users are only allowed to query the tags they have already paired, through conventional Bluetooth, with

their smartphone. Other services, such as Locus Pocus [21], are device-agnostic and enable the user to track any BLE device and tag. Locus Pocus, instead, monetizes the offered service by charging a fee for querying the location of tags. Besides general purpose tags that can be attached to different objects, BLE beacons are also being incorporated in various commercial products, like luggage [2], for tracking purposes. Automotive companies have also started deploying vehicles with support for such functionality, with Land Rover integrating support for Tile [52].

The success of services that build on crowd GPS mainly depends on two factors: battery life and user base. BLE tags are powered by small batteries, so energy efficiency is critical to keep the maintenance cost of battery replacement low, while the number of people participating in a crowd GPS crowd determines its coverage. Consequently, the existence of multiple, disjoint GPS crowds, created by different vendors hampers their effectiveness. As Techu is not tied to a specific type of tag and is built around a public server, our approach can reduce the segmentation of the user base.

2.2 Privacy Threats of Crowd GPS

Despite the unprecedented opportunities offered by the paradigm of crowd sensing, which can have significant societal impact in critical scenarios, these technologies do not come without risks for participants. Alarmingly, in scenarios where users frequently expose their location to the service, that information can be leveraged for inferring sensitive information [54] (e.g., medical conditions, religious beliefs, sexual orientation) or deanonymizing the user [49]. In the context of object-discovery services, the privacy of users is also threatened by the periodic broadcasting of information from the tags, which are usually carried by the users, as it can lead to tracking [45]. Since the utility of many services stem from the aggregation and processing of location-aware data from users, in practice, participants can be subjected to tracking. This can be catalytic in deterring users from participating. For example, both TrackR and Locus Pocus mention that the collected data, including location, may be shared with third parties. The latter also explicitly states that data may be shared with law enforcement agencies.

The current state of *fragmentation*, with each tag vendor deploying a platform for their own set of tags, significantly reduces the effectiveness that could be achieved by a vendor-agnostic platform that unifies all disparate crowds into a single mass. Furthermore, the existence of multiple centralized proprietary services also increases the chances of misuse. Given the frequency of security breaches and theft of user data [5, 6], centrally-stored user location information poses a significant privacy risk. As we demonstrate in § 3, the security flaws of TrackR enable the tracking of their users.

Our goal is to highlight the privacy and security concerns that stem from device tracking in this emerging ecosystem, which motivates our design of a decentralized public crowd GPS platform, that in not bound to any specific type of tag and *does not require any manufacturer modification.*

3. A SECURITY ANALYSIS OF TRACKR

Here we present our security analysis of TrackR, one of the most popular crowd GPS services, which has also partnered with Amazon [7]. While claims of being the "largest Crowd GPS network in the world" [9] cannot be verified, the reported number of over 5 million users across the USA and Europe renders TrackR a suitable candidate for highlighting the significant privacy risks of crowd GPS.

We obtained the official TrackR app for Android from the Google Play store and decompiled it, following an established procedure. This involved unzipping the .apk file, using `dex2jar` to convert the extracted .dex files to class files in a .jar archive, and finally decompiling the jar file using `jd-gui`. The retrieved code was readable almost in its entirety. Review of the app's code revealed that the app communicates with TrackR servers through `RESTful` calls. As expected, the TrackR app scans for BLE beacons of the vendor and uploads their IDs (i.e., their Bluetooth MAC addresses) to TrackR servers along with the GPS coordinates where they were seen. The apps looks for TrackR's manufacturer ID in the received BLE packets, so the crowd only tracks beacons of the vendor. To search for a lost tag, the app issues a call to the service using the following URL:

https://phonehalocloud.appspot.com/rest/tracker/[tracker-id]

The tracker-id is not the beacon's MAC address (BSSID), but a value derived from it, which is calculated by appending the string representation of BSSID, least-significant byte first, to '0000'.

For instance, a tag that has the address `01:23:45:67:8a:bc:ef` will have the tracker-id `0000efbc-8a67452301`. The service responds with a JSON object, similar to the one shown in Listing 1, containing information about the most recent observation of the tag, including a timestamp, coordinates, and its battery level.

Tracking Vulnerability. While most of API calls of the TrackR service require authentication (e.g., an OAUTH token), the URL above, used to search for lost tags *does not*. As such, *anyone* can at *anytime* query about *any* tracker they have encountered at least once. As a result, unauthorized parties can locate tags and track the owner. By periodically obtaining this information adversaries can reconstruct user trajectories, which can lead to their identification [24, 32] and the inference of sensitive information.

Ethical Considerations. Upon discovering this vulnerability in April 2016, we immediately disclosed it to the vendor and offered a detailed report of our analysis. Some might argue that disclosing an attack against a popular service raises ethical concerns, as this attack can be deployed in practice by malicious individuals. However, we believe that it is crucial to raise the public's awareness and expedite patching, as millions of users are already participating in such services, and numbers are expected to grow even more in the near future. Nonetheless, we have followed the widely accepted practice of responsible disclosure, and offered a detailed report of our analysis to the vendor in an effort to help improve their system.

As a final note, it is important to highlight that despite the straightforward nature of the attack and the corresponding fix, it took TrackR more than 9 months (December 2016) to address this issue. To make matters worse, in the meantime, they introduced new attributes in the returned `JSON` message that leaked more information as can

```
{"lastTimeUpdated": 1460751872294,
"timeUpdatedDiff": 256354145,
"lastKnownLocation": {
"longitude":-73.98,"latitude":40.75},
"trackerId": "0000EFBC-8A67452301"}
```

Listing 1: Example response of tag tracking API call at the time of disclosure (April 2016).

```
{"lastTimeUpdated":1467043953079,
"timeUpdatedDiff":3307969283,
"lastKnownLocation":{"longitude":-74.03,"latitude":40.74}
"friendlyName":"**","locationName":"***"
"trackerId":"0000EFBC-8A67452301"}
```

Listing 2: Example response of tag tracking API call at a later time (August 2016), leaking additonal information.

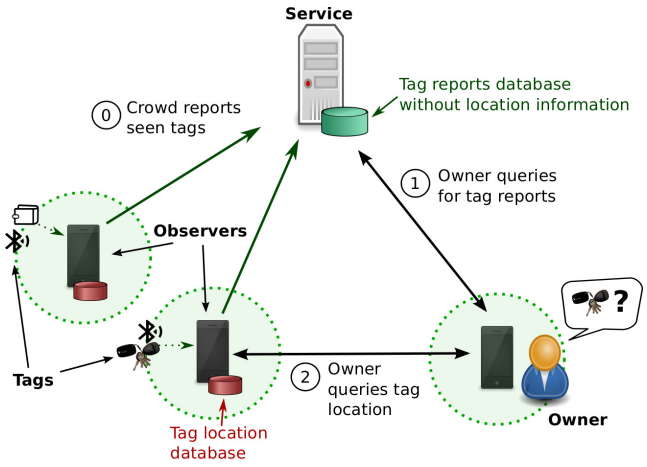

Figure 1: Design overview. The crowd regularly sends reports of observed *tags* (without location information) to the *server* ⓪. When tagged property is lost, the *owner* queries the server to learn who *observed* the tag ①, and contacts them to obtain the corresponding locations ②.

be seen in Listing 2; the one denoted as "friendlyName" reveals the user-supplied nature of the tagged item.

4. TECHU: CROWD GPS

Threat Model. To better capture the privacy threats that crowdsourced GPS systems face, we consider the server component to be untrusted and assume a "*malicious but cautious*" behavior, which has been proposed for cloud computing providers [20]. In this model, the server can actively attempt to learn or deduce information from incoming reports and queries, but will try to appear legitimate by returning the appropriate results to queries, as users could easily identify records being tampered with. We do not require users to register or exchange any keys with the server, and assume that they can obtain a non-tampered version of our app, e.g., through the official app store.

Our threat model captures two orthogonal adversarial behaviors that target complimentary aspects of Techu. In the first case, the adversary's goal is to compromise the privacy offered by design, and obtain sensitive information about the user(s). This may include tracking a user's location through time, or inferring sensitive information and uncovering the user's identity by obtaining location data (e.g., home or work address [30]). In the second case, the adversary is driven by economic incentives, and monitors the information available so as to identify lost tags and attempt to recover them. We assume that adversaries have access to all information contained in the bulletin board, can deploy a *limited* number of physical "agents" within the crowd, and even report fake events to the system. Furthermore, the adversary can monitor traffic incoming to the server, and has significant computational resources at his disposal, thus, is able to perform brute force attacks against information encrypted with low entropy. As is the case with all crowdsourcing services [11], our system is susceptible to users reporting fake data (e.g., spoofing their location [59]).

4.1 System Design

We designed Techu as a decentralized privacy-preserving crowd GPS system that addresses the privacy risks of existing vendor-based solutions and services proposed in the literature. Our design was motivated by the inherent risks of collecting location data centrally, including data leaks, cryptanalysis attacks against encryption algorithms, and weak keys. We further discuss the security advantages of our approach over a centralized design in § 5. Our system consists of four entities: the *tags*, the *owners* of the tags, the *observers* that form the crowd, and the *service* that collects reports from observers and accepts queries from owners.

A high-level overview of our design is depicted in Figure 1. Owners store an ephemeral ID on each of their tags, which is unique for each of their tags and is periodically updated when in range. Tag IDs are broadcast to nearby observers through BLE advertisement packets and become part of the reports observers send to the service for nearby tags (step ⓪). Reports do not include location data, but contain sufficient information for owners to identify the reports that correspond to their tags and contact the observer that issued them. When a tag is lost, owners first retrieve the corresponding reports for a desired time range from the service (step ①) and then proceed to contact observers and retrieve the location where a tag was seen, after proving to them that they are the tag owner (step ②). Below, we elaborate on the various phases of our system's functionality.

Generating tag IDs. For each tag, owners periodically generate a key pair (pk_i, sk_i) using an asymmetric-encryption algorithm, such as ElGamal, Elliptic Curve techniques, or RSA, with the public key, pk_i, acting as the tag's ID, t_i:

$$t_i \equiv pk_i$$

t_i is stored on the tag using a writable BLE attribute, which is accessible only to the owner after pairing with it. Unauthorized users should not be able to overwrite it, as, among other issues, it would disrupt the operation of the system. To make it observable to the crowd, t_i is included in the advertisement packet periodically transmitted by the tag to announce its presence. In commercially available tags this can be done by storing t_i into the `Device Name` attribute. Tag IDs are *ephemeral* and *unlinkable*, that is, we periodically update them and each ID is independent of the others. As such, an ID t_i is only associated with a tag for a limited amount of time, preventing long-term tracking of the tags, since third-parties cannot guess t_{i+1}. Techu does not use MAC addresses to track tags. Instead, we assume that tags are using private MAC addresses [18] (further discussed in § 8) to avoid tracking from malicious observers [45].

Reporting observed tags. Each observer in the crowd is identified by $observerID_k$, an ephemeral client ID in a messaging service, like Google's Cloud Messaging (GCM), which the owners use to contact them. For each tag observed, they upload a report r_j to the service, where:

$$r_j = ts_j \mid E_{pk_i}(observerID_k) \mid \hat{t_i}, \quad \hat{t_i} = H(t_i)$$

Report r_j includes the timestamp (ts_j), the observer's ID encrypted with the public key pk_i (i.e., the tag ID t_i), and the secure hash digest of t_i ($\hat{t_i}$). The encryption function E actually creates a digital envelope utilizing a secret-key algorithm and message authentication to provide confidentiality and integrity. This ensures that only the tag owner can obtain the observer's ID from a report, and only the owner or an observer of a tag can identify the reports associated with a particular tag, because they can generate $\hat{t_i}$.

Storing tag locations. The actual location of the observation is stored locally by observers, as tuples of:

$$\{ts,\ t_i,\ location\}$$

We assume that the service has enough storage to store reports for a few days or even weeks to facilitate the recovery of lost property. On the other hand, the storage required by observers can be bound by discarding old observations, keeping only the most recent entry for each tag, or even probabilistically dropping certain entries.

Querying lost tags. To retrieve information for a lost tag, the owner issues a query requesting all the reports corresponding to

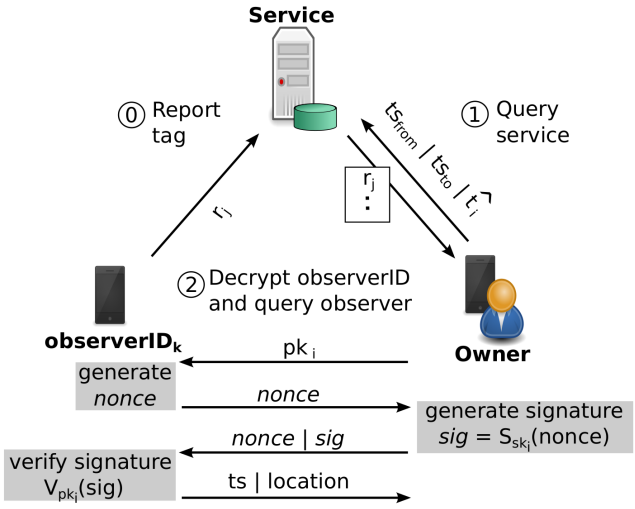

Figure 2: Messages involved in tracking tags and querying for lost tags.

a tag's $\hat{t_i}$ for a time range (ts_{from} : ts_{to}), which are returned by the service , as shown in step ① of Figure 2. Since queries are performed based on $\hat{t_i}$ and a time range, they are lightweight. By choosing a hash function like SHA-3, collisions are highly improbable and the returned reports will most likely correspond to the owner's tag.

For each returned report r_j, the owner decrypts the observer ID information, using the tag's secret key sk_i. The owner then communicates with observers over a channel that ensures confidentiality, e.g., over TLS, to prove ownership of the tag and obtain the location information. Various techniques can be used to achieve this, but we employ the well-established technique of digital signing, as shown in Figure 2 (step ②). The protocol is as follows: the observer generates a random number $nonce$, which it transmits as a challenge to the owner. The owner signs the nonce using function S and the appropriate private key and sends it to the observer. The observer can verify the validity of the signature using function V and the public key of the tag t_i. If verification succeeds, the location information can be sent to the owner. pk_i is used to search the locally-stored data and, in the case of a match, respond with the timestamped location of the tag.

4.2 Privacy Extensions

In this section, we present two extensions to our basic design that trade performance for improved security and privacy properties. We evaluate the tradeoffs in § 7.

4.2.1 Decoy Queries

We introduce the use of *decoy* lost-tag queries, which are sent to observers so as to obfuscate actual incidents of owners searching for lost tags. This creates "uncertainty" which deters malicious observers from reasoning about tags being lost, as the received query could be a decoy. Owners initiate the process at random intervals emulating the loss of one of their objects and its tag. The whole process needs to be indistinguishable from a real tag discovery event, so it follows the exact same steps, i.e., the owner retrieves the most recent reports from the service (e.g., the ones sent in the last hour) and it issues queries to the corresponding observers.

We choose to generate decoys at random intervals, even though, one might argue that in the real world it is unlikely for inquiries to occur during the night, when most users are asleep. That would allow malicious observers to infer that a request received at night has a higher probability of being a genuine one. In other words, maximum uncertainty is achieved when all events are equally likely to happen [64]. To generate decoys, each owner periodically (with period T) decides to initiate a decoy event with probability p.

Security improvements. Observers that receive queries from owners are now uncertain of whether the query corresponds to an object that was actually lost. In particular, if an owner initiates on average X decoy lost-tag events per hour, where $X = T \times p$, and an observer has reported N tags belonging to different owners, then the observer will receive $X \times N$ decoy queries per hour. The probability of an observer receiving a real tag discovery query is probabilistically very small; our goal is to hinder malicious observers from "scavenging" for lost items they receive queries for. Essentially, we aim to maximize $X \times N$ without overburdening benevolent observers, which could lead them to blocking owner queries or abandoning the system. For the same reason, which tags are queried in the decoys is not of importance and could be selected randomly. This extension allows owners to insert a configurable amount of noise in the system that will render scavenging unprofitable.

4.2.2 Salted Tag IDs

Tag tracking. We can replace the hashed tag ID $\hat{t}_i$ that is sent to the service, with a salted version $\dot{t}_i$. For each report, we generate a random number, $salt_j$, and hash it with t_i:

$$\dot{t}_j = H(salt_j \mid t_i)$$

To facilitate querying the server, we generate a "color" value c_j consisting of the b least-significant bits of the unsalted hash value $\hat{t}_i$. Since we only keep a few bits of the hash, c_j is not a unique value, and reports for different tags will share the same color. So given a function LSB that keeps the b least-significant bits, the color is produced as:

$$c_i = LSB_b(\hat{t}_i)$$

The report $\dot{r}_j$ is extended to include the salt and color:

$$\dot{r}_j = ts \mid o_i \mid \dot{t}_j \mid salt_j \mid c_i$$

As before, an HMAC code is appended to the report.

Querying lost tags. The owner uses t_i to produce a c_i to query the service. Since different tags will be colored the same way, reports referring to other tags will also be returned. A time range (ts_{from} : ts_{to}), which remains part of the query, can be used to reduce the number of reports retrieved from the service. For each report $\dot{r}_j$, the tag owner uses the tag's t_i and the salt ($salt_j$) contained within the report to produce $\dot{t}_j$ and compare it to the one in the report. Matching reports correspond to the owner's tag, and are used to query the observer for the tag's location as before.

5. SECURITY ANALYSIS

Techu is designed to ensure the location privacy of the crowd's participants, while offering more effective object tracking as opposed to other proposed systems [28, 70] that attempt to locate the tags after the issues are queried. Here we present threats that affect crowd GPS systems, and outline the privacy guarantees that are attained by core aspects of Techu's basic design (§ 4.1) and privacy extensions (§ 4.2).

Untrusted Server: owner privacy. In our design, the service cannot obtain any information regarding the location of tags, as that information is stored locally by the observers. As that information is only disclosed to the owner *after proof of ownership* through the private key *pk*, the server cannot trick the observers into divulging the information.

Untrusted server: observer privacy. The service cannot learn anything about observers because their IDs are encrypted using t_i, which is never disclosed to the service by the owner nor any observers. The service could potentially try to estimate an observer's IP-based geolocation. However, previous work has demonstrated that geolocating IP addresses that belong to cellular networks cannot achieve accurate results; using geolocation databases results in errors in the order of tens of kilometers [72], while measuring application-level latencies can only be used to distinguish cities [14]. As such, both of these techniques would fail to offer any useful location information to an adversary.

Malicious Observers: *direct* tag tracking. By using ephemeral IDs for the tags, we ensure that observers cannot continuously track tags based on their t_i. In the case of observers colluding and forming a malicious crowd, such as a malnet [44], observers can track the tag after they observe it for the remaining time window (e.g., a few minutes), until the generation of the next t_i [2]. This is not long enough to reconstruct a trajectory of the user's whereabouts, and is the same trade-off faced by existing privacy-oriented mechanisms like private MAC addresses. Thus, Techu offers significant location privacy guarantees over previous systems that were susceptible to user tracking [22, 70].

Malicious observers: *indirect* tag tracking. After logging a device, observers can use the tag's ID to query the service and obtain the observation events generated by other observers for that ID while it was active. However, observers cannot obtain location information from other observers because they do not possess the secret key sk, thus, lack the ability to prove ownership.

Malicious observers: co-observer privacy. The observation events uploaded to the service contain the observer's ID encrypted using t_i. This prevents users from discovering the IDs of other observers that reported the tag in their vicinity. Furthermore, these IDs for communicating with apps over Google's GCM are in no way linked to the user's actual identity, and do not disclose any actual information about the user. Moreover, these IDs are periodically changed, to prevent third parties from linking the same observer across different observation entries.

Malicious observers: scavenging lost tags. Observers in our system may (try to) infer that a tag was lost, and attempt to retrieve/steal the lost property the tag corresponds to.

Tag discovery query. When observers receive a discovery query from the tag's owner, they can directly infer that the tag has been lost. We address this threat with our *decoy query* privacy extension. Observers that receive queries from owners are now uncertain of whether the query corresponds to an object that was actually lost. The goal is to maximize the number of decoy queries without introducing a significant overhead for observers in terms of battery consumption, which could result in users abandoning the system. While introducing noise in the system results in uncertainty and renders scavenging unprofitable, nonetheless, in practice attackers could still attempt to physically recover any tags for which they received queries (including potential decoys).

Tag ID persistence. Owners periodically change their tags' name to prevent observers from being able to track their location. As a side-effect, once a tag is separated from its owner it will retain the same t_i for a longer period of time, allowing observers and the service to infer that a tag may have been lost (or purposefully left behind by the owner). While the time window is configurable, thus allowing each user to tweak the tradeoff between privacy and power consumption based on their own requirements, our system sets an upper bound of 10 minutes to ensure the privacy of participants.

[2]Users can configure the time window to be arbitrarily short.

5.1 Alternate Design Analysis

Here, we present an alternative centralized design and the inherent risks it suffers from. While such a centralized approach simplifies the design and reduces the overhead, it also introduces significant privacy risks. Insider threats [15], the interest of intelligence agencies in tracking users [10], the precedence of law enforcement agencies issuing subpoenas to access user data [67], and the frequency of data breaches [5], all highlight the risks of centralized crowd GPS systems.

Design. In the centralized design, observers upload reports containing a timestamp (ts), the observed tag's location encrypted with t_i as the public key, and the secure hash digest of t_i. Owner's then simply fetch the most recent reports with a hash digest that corresponds to their tag's ID, and decrypt the location using their private key.

Cryptanalysis. All reports uploaded to the server can be trivially collected and stored by adversaries. Even though the sensitive information (i.e., the *location*) is encrypted, previous work has extensively explored the risks of archiving encrypted sensitive information and the need for long term secrecy (e.g., [15, 68, 69]). Specifically, Storer et al. [69] argued: *an adversary who can compromise an archive need only wait for cryptanalysis techniques to catch up to the encryption algorithm used at the time of the compromise in order to obtain "secure" data.* In this case, there is no need for compromising the archive, as it is publicly available online.

To make matters worse, the hardware restrictions of BLE tags limit the potential key sizes, further increasing the risks of storing the sensitive information on the server. While current recommendations call for minimum key sizes of 256 bits in Elliptic Curve cryptography [1] (which has already been deprecated by the NSA), BLE advertisement are restricted to 248 bits per the BLE specification which allows a maximum of 31 bytes used as data payload [17]. This size restriction naturally leads to the use of elliptic curves instead of other public key techniques like RSA. While more data can be transferred through a scan response packet, that would incur a massive overhead as it would require an extra packet sent for each device in the tag's vicinity. Furthermore, as the size of the advertisement payload impacts the tag's battery life [62], further increasing the advertisement packet size could have undesirable implications, given the low energy consumption tenet of BLE technology. Finally, notable cryptographers have expressed concerns regarding the robustness of elliptic curves against potential advancements by the NSA [35, 48]. Assuming that the adversary decrypts the encrypted locations in the report, we describe two attack scenarios that can recreate users' location traces.

Attack 1: observer tracking. A malicious but curious (or compromised) server can link observer reports uploaded from a specific IP address. As users typically connect over a cellular network when on the move, attackers could recreate a user's daily commute and identify the user [30], and infer sensitive data from other visited locations. To gauge how persistent cellular IP addresses are, we conducted an experiment over the duration of one week with 3 devices, that belong to the authors, connected to a major US cellular provider. We found that the IP address changes when the cellular connection is switched off (e.g., switch to WiFi, turn off device) or if the cellular signal is lost due to bad reception; in our experiments IP addresses remained unchanged for 3-62 hours when the connection was not manually switched, demonstrating the practicality of such an attack.

Attack 2: tag tracking. While the ephemeral tag IDs used in Techu prevent the straightforward tracking of tags, a malicious server could potentially link successive ephemeral IDs. Figure 3, presents an example scenario where four participants are in the same area.

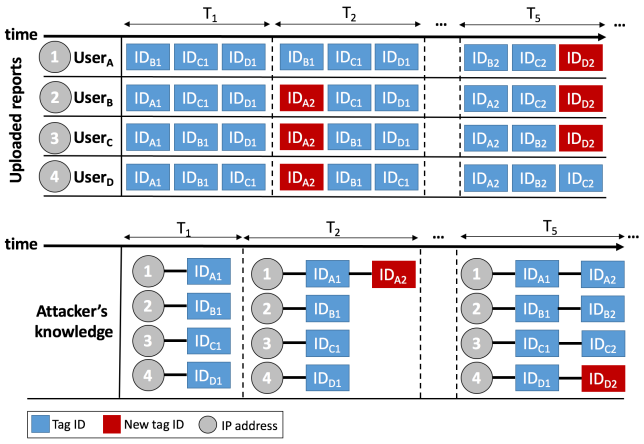

Figure 3: Example scenario with 4 co-located users, outlining an attack that links users' consecutive tag IDs.

Note that the different time windows T_i do not have the same duration, but only represent the time window within which all tag IDs remain the same. For instance, during T_1 every observer uploads reports regarding the remaining 3 tags in the vicinity. Based on the IP address of the observer uploading each report, the adversary learns which tag belongs to the user behind each IP address. Now, suppose that the first user's tag gets a new ID assigned, which means that we are now in T_2; since the first user will report observations for the same tags as in T_1, while all other users will report 2 identical tags from before plus the new tag, the attacker can now associate the new tag ID_{A2} with tag ID_{A1}. This can similarly be done for all the tags. Its important to note that this is a hypothetical scenario that assumes that the set of users remains constant for some time, and in practice this assumption may not hold true. The dynamic nature of user mobility and scenarios where large crowds of users momentarily coincide may introduce significant complexity that hinders the practicality of this attack. As such, while various settings seem suitable for such an attack (e.g., mass transportation rides between stops), further exploration is required for evaluating the feasibility and effectiveness of the attack under realistic conditions.

Motivation. All these issues motivated the *decentralized* design of Techu where location information is *temporarily* stored by observers and shared only upon *proof of ownership*, thus, nullifying the threat of these attacks.

6. IMPLEMENTATION

In this section, we present details regarding our prototype implementation, and discuss aspects of BLE, the more energy-efficient version of Bluetooth; BLE was developed for communicating with devices with limited resources, and presents constraints that guided the design of Techu.

We implemented a prototype for the client-side software of observers and owners on the Android platform, and built a service for collecting the data from observers as a web application over JBoss AS 6.3. The clients handle BLE devices following the Bluetooth specification v4.2, which defines that advertising packets can have a payload of 6 to 37 bytes. The first 6 bytes are reserved for the device's advertisement address (i.e., the Bluetooth MAC address). The remaining 31 bytes can be used by developers to store the data that will be advertised by the beacon to the other devices. More information on how those bytes are broken down into commonly used data types can be found in the specification [18].

Experiment Location	Duration	tags per hour	
		Average (std.dev.)	Max
Large US city metropolitan area	41 days	57.33 (52.73)	491
Large US city center	4 hours	274.67 (162.59)	549

Table 1: Results from two experiments where BLE tags were collected in the wild.

Due to the limitations of the Bluetooth specification, we can advertise tag IDs of up to 248 bits through the advertisement packets. We used AES-128 for symmetric crypto, SHA3-224 as a secure hash-function, and elliptic-curve ElGamal with 224-bit public keys for PK-encryption. We used the security providers[3] bundled with our device when available, and the Bouncy Castle[4] library v1.54 for ECC.

For owners to communicate with observers that have reported seeing their tags, we utilize the Google Cloud Message (GCM) service. Our design deviates from the common GCM architecture as our app registers 2 different IDs and runs both as a server and a client; one serves as the user's `ObserverID`, and the other is used for sending tag-discovery queries to other users (preventing observers from learning a tag owner's `ObserverID`). Crowd clients communicate by pushing XMPP messages to the GCM service. Clients periodically (e.g., every few days) may obtain new IDs by creating new instances and re-registering with the GCM service; this enables the ephemeral nature of observer IDs. While new observations will be reported under the newly acquired ID, by keeping the older instances alive, up to a given time, observers can receive queries for tags reported with the previous `ObserverID`. As this is not the intended use of GCM, in the future it could be modified to invalidate the old IDs, or limit the amount of instances each client may register. While this would prevent the use of ephemeral observer IDs, the privacy offered by our system would remain. We would like to emphasize that there is no app or user registration server in Techu, which could be potentially breached.

TrackR's Bravo tags expose a writable `NAME` attribute that we use for storing the tag's ID t_i, as its contents are placed in the payload of the advertisement packet. This behavior is typical for BLE devices. For example, we also tested Jawbone UP3 fitness wearables and found they expose a name attribute which can be written and is advertised.

7. EVALUATION

We evaluate Techu in terms of the overhead it imposes on owner and observer smartphones, and estimate the storage requirements for the server. For our experiments, we used TrackR Bravo tags, and two stock, low-to-midrange smartphones as observers and owners: an Asus Zenphone and a BLU Studio Energy 2, both running Android 5.0.

Tag density. The effectiveness of crowd GPS systems depends on the size of the crowd. While there are projections for an explosion in the number of BLE devices [73], to the best of our knowledge, there have not been any studies about the number of devices that can be currently found in the wild. To help us establish the parameters and quantify the overhead of our system if deployed today, we performed two experiments collecting tags. We present the summary of their results in Table 1.

[3]I.e., the classes extending the java.security.provider class.
[4]https://www.bouncycastle.org

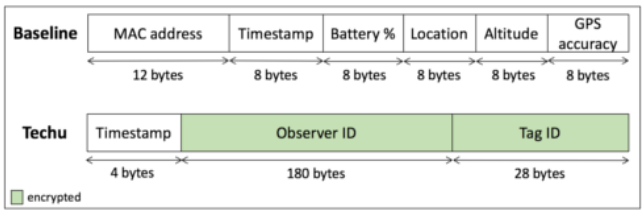

Figure 4: Comparison of reports sent by a baseline crowd GPS and Techu's tag observations.

	Average	Std. dev.	Median
Baseline	564ms	134ms	524ms
Techu	1025ms	140ms	1011ms

Table 2: Report transmission times including required cryptographic operations, for batches of 50 reports (based on tags/hour observed in mid-density area).

In the first experiment, we continuously collected BLE tags using the smartphone of one of the authors for 41 days, in the broader metropolitan area of a city; the majority of measurements were collected in a mid-sized town where one author works (Hoboken, NJ), including occasional trips to the city (New York City, NY). The dataset includes measurements from workdays as well as weekends. In the second experiment, we included measurements from a four hour period (4-8pm) collected during a weekday in the busiest center of a major metropolitan area (Manhattan, NY). In both cases, we took measurements every 5 minutes and identified unique BLE devices using their MAC addresses. Unsurprisingly, the results show that in a *high-density* area (e.g., the city center during peak hours), we observed significantly more BLE tags on average, than in a *mid-density* area. We use these observation rates to frame the remaining experiments.

Reporting overhead. Reporting the observed tags imposes an overhead on observers in terms of both storage and battery consumption. In this section, we evaluate the overhead of Techu compared with a commercial crowd GPS service such as TrackR, referred to as the baseline. Figure 4 compares the reports sent by Techu with the baseline. Even though we do not transmit location information to the service, due to the size of the *encrypted* observer ID (180 B), we need to transmit about four times more data for each reported tag. Transmitting these reports to the service is not four times slower though, because even when batching reports together the overhead of establishing a connection over SSL dominates the cost. Table 2 shows the time needed to transmit 50 reports (the average tags/hour observed in a mid-density area) over WiFi to the service.

	High-density Area		Mid-density Area	
	Rows	Size (KB)	Rows	Size (KB)
Hour	275.00	10.00	57.00	2.20
Day	6,592.00	258.00	1,368.00	53.40
Week	46,1447.00	1,803.00	9,576.00	374.10

Table 3: Estimated size of client database when retaining data for different time periods.

Techu stores the location of recently observed tags on the client side. In Table 3, we show the number of reports and amount of storage required for retaining location information for an hour, day, and week, for high and mid-density areas. Our prototype requires

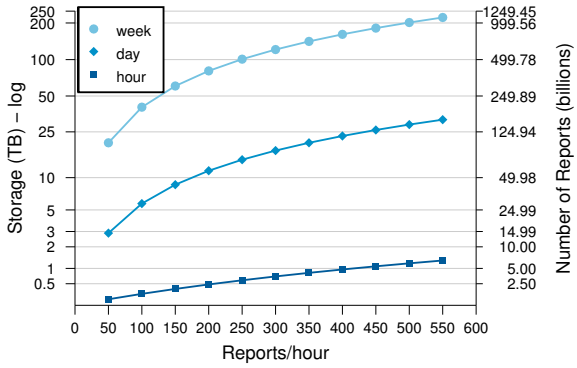

Figure 5: Storage requirements per data retention period, based on the reports/hour generated by observers, under an extreme scenario of 12 million users in a city.

40 bytes per report (tag ID, timestamp, location), resulting in negligible storage requirements, even when retaining reports for a week.

Battery overhead. Preparing and transmitting the reports to the service also consumes energy. For Techu this includes all cryptographic operations and transmissions. For the baseline crowd GPS, we simply transmit reports to the service. As both the baseline and Techu need to activate the BLE adapter to receive advertisement packets, we focused our experiment on the differentiating aspects of our approach. We measure the impact on the battery by sending 1000 reports to the service, which results in a 3% drop in the battery level for Techu, compared to 1% for the baseline. While the relative overhead seems non-negligible, we should emphasize that if we periodically send reports to the service every hour, 30, 5, and 1 minutes, 1000 reports will be sent in 41.6, 20.8, 3.5, and 0.7 days respectively.

Service overhead. Figure 5 shows the server's storage requirements. We plot the estimated size versus the reports/hour generated by observers, using a broad range that captures both mid and high-density areas. We assume a city size of 12 million users (significantly larger than New York, the most populous city in the US), who are all part of Techu and continuously report at the same rate. Assuming typical mobility behavior, our system will require 23 TB of storage for retaining the reports of the past week. At the most extreme scenario where the entire population forms high-density clusters, storage may surpass 200 TB. That, however, is an unrealistic scenario and in practice requirements will be significantly lower than 23 TB even for major cities.

Querying lost tag. While this operation is infrequent and only occurs when an owner attempts to locate a lost tag, it is critical since a high latency between querying for a tag and receiving an answer might render our system ineffective. We tested three different connection setups to evaluate performance over all possible connection combinations between the owner querying about the tag and the observer that holds the location data. We issued 1000 requests from a device acting as the owner to a device acting as the observer, and measured the time required to receive the location information for the observed tag. As shown in Figure 6 the cost is less than 2 seconds in the worst case scenario, which occurs when both the owner and observer connect over 3G.

Overhead on tag. Our design periodically writes a new tag ID to the BLE tags, and this additional operation may impact the tag's battery life. Experimental measurements from other sources [47, 62] and the BLE specification indicate that the cost of a write operation is on par with the transmission cost of an advertisement

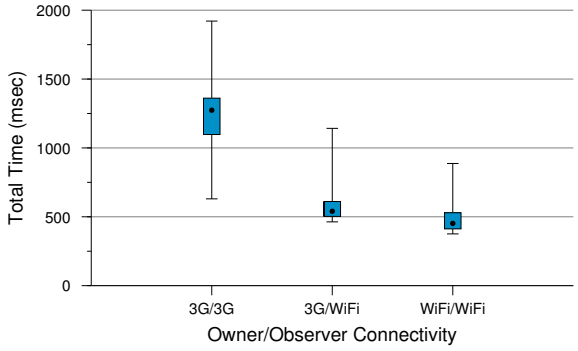

Figure 6: Time required to query the location of a tag from an observer, depending on type of connectivity. Median denoted by the "dot" commonly reffered as Q2, the bottom side of the box is Q1 (lower median), the upper side of the box is Q3 (upper median), the whiskers show the min and max value respectively.

Write Period (min)	Lifetime (days)	Lifetime Reduction (%)
1	35.82	8.83
5	38.54	1.90
10	38.91	0.96
20	39.10	0.48
30	39.16	0.32
60	39.22	0.16
120	39.25	0.08

Table 4: Tag battery lifetime reduction due to frequency of writing new IDs (normal lifetime is 39.29 days).

package for small payloads. Since our design writes the tag at a frequency that is as least two orders of magnitude *lower* than the frequency of broadcasting advertisement packets (e.g., once a minute vs 100-500ms), we did not expect a considerable impact on battery life.

We experimentally verified our expectations by performing over 100,000 writes to a TrackR tag over the course of a week. Simultaneously we activated a second tag as a control tag. We then measured the battery voltage of both and used the manufacturer's data sheet [53] to estimate the remaining mAh. Using our control tag we calculated the difference due to our write operations, as:

$$\Delta(C_{control} - C_{test})/C_{full}$$

where the nominator is the difference in remaining charge between control and test tag and the denominator is the capacity of a brand new battery. The battery model was CR1616 with a charge of 55mAh when full. The results for different frequency updates are shown in Table 4. Even for tag generation periods as small as 5 minutes, our system reduces the lifetime of the battery by less than 2%, which is negligible compared to the privacy benefits obtained.

Salted tag IDs extension. When using salted tag IDs (see § 4.2.2), the owner needs to fetch a larger part of the database, which is determined by the color c_i of the tag. Once those reports are obtained, the owner much check each returned report to determine if it corresponds to the desired tag. This requires calculating a hash for each report using the salt contained in the report and the tag's ID. The number of bits used for c_i, along with the time range used in the query, determine the number of entries returned to the owner.

In Table 5, we show the number and size of reports returned to the user, along with the time required to produce a hash for each report, when using different numbers of bits for c_i and querying for data from the last hour, day, and week. We notice that by using 16 bits (out of 28) of the tag ID as color, an owner querying for a tag

c_i size (bits)	Query Range	Downloaded Reports		Generating Hashes (sec)
		Size (MB)	# (in thousands)	
8	Hour	5,630.40	25,781.00	3,532.21
	Day	135,129.69	618,750.00	84,773.08
	Week	945,907.83	4,331,250.00	593,411.57
16	Hour	22.09	101.00	13.80
	Day	530.16	2,417.00	331.14
	Week	3,711.09	16,919.00	2,318.01
24	Hour	0.09	0.39	0.05
	Day	2.08	9.44	1.29
	Week	14.56	66.09	9.05

Table 5: Size and processing effort for discovering location of lost tag when using salted hashes, for maximum number of generated reports/hour (taken from Table 1).

	Basic	Salted IDs (Color bits)		
		24	16	8
Mid-density	0.01 MB	0.02 MB	3.61 MB	938.40 MB
High-density	0.07 MB	0.09 MB	22.09 MB	5,630.40 MB

Table 6: Data fetched from service for query targeting one hour time window, under different configurations.

lost in the last hour will need to download approximately 22 MB of reports and spend less than 15 seconds processing them. When using a 24 bit color, even for a query window spanning an entire day, the owner would only need to fetch about 2 MB. Note that the reports stored in the service now include an additional 8-byte salt and a color value of variable size.

Decoy queries extension. Decoy queries (see § 4.2.1) defend from malicious observers that wish to identify and steal lost objects. However, a high number of decoys introduces overhead. To quantify this, we consider a scenario where $N + 1$ users all own one tag and, within a certain time window, all observe every other tag[5]. Each user has two roles:

- *Owner:* randomly decides to issue decoy query to each of the N observers of his tag.
- *Observer*: receives N decoy queries for observed tags.

If every member decides to issue one decoy query within the time window, each user will need to query the service once and participate in $2 \times N$ queries, as an owner or observer.

Table 6 shows the amount of data (in MB) that need to be fetched by owners for various configurations of Techu in the two tag-density scenarios we have been considering. We notice that, when using salted IDs, unless we use a large color value, decoy queries become expensive for tag owners.

In order to calculate the average energy cost of performing a single query, we configured our testbed smartphones to exchange messages over 3G and performed 1000 queries. Our experiments show that the cost of issuing and responding to query are the same, hence, the overhead is mostly affected by the frequency decoys are generated at.

Figure 7 shows the number of queries/hour that would be generated by a user, for different decoy-generation frequencies in the mid and high-density areas. It also shows the battery consumption inflicted on the user, who acts both as an owner and observer (i.e., both sends and responds to decoy queries). If owners issue one decoy query per hour on average, in the medium-density area an observer receives and issues 57 decoy queries every hour. Our results indicate that we should adapt the frequency decoys are gen-

[5]Obviously this offers an extreme upper bound and, in practice, the introduced overhead would be orders of magnitude lower.

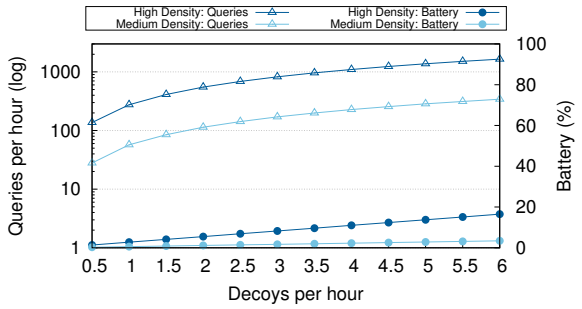

Figure 7: Cost of decoy queries extension for the two different population density datasets.

erated at based on the density of the area, as it could otherwise lead to a large number of decoy requests and significant overhead.

8. DISCUSSION AND FUTURE WORK

Geolocation. As aforementioned, IP-based geolocation techniques for cellular connections do not offer enough accuracy to pose a risk for users. If that changed, users could employ Tor [25] or VPNs for hiding their IP address; that, however, would impact the usability and adoption of Techu.

Beacon Sampling. Our experimental evaluation revealed the moderate storage requirements of the server, even when deployed for major metropolitan areas. As we envision Techu being deployed as part of community effort and not by a private vendor, maintaining state-wide "instances" of our service may require significantly more storage, which could potentially hinder such initiatives. Since in high-density areas many of the reports are overlapping (i.e., concern the same tag), we can reduce the volume of reports by probabilistically sampling the beacons to be reported. We leave the analysis of such an approach as part of our future work.

Silent Zones. An adversary with significant physical resources, could distribute a large number of malicious tags, and attempt to track users through their observer IDs. Alternatively, adversaries that have collected the reports over a long period of time and manage to decrypt them, may try to infer very coarse-grained information regarding a tag's location, based on the number or frequency of observation reports for specific tags. Through empirical measurements, adversaries could potentially differentiate locations based on the density of observers at specific locations. This could allow the adversary to reconstruct a user's trajectory and infer the user's work and home locations; previous work has shown how that information can lead to the identification of the user [30]. Our *salted tag ID* privacy extension significantly increases the difficulty of such an attack, as the adversary would have to decrypt a large number of reports (if not all), to obtain correct number/frequency information per tag.

While these attacks would require significant physical resources, nonetheless, users can configure *silent zones* within which the Techu app will not search for tags or create observation reports. These areas need not be large, which could also impact Techu's effectiveness, but would merely contain the two city blocks containing their workplace and home. Silent zones could potentially pose a privacy threat if the user's whereabouts are tracked over a very long period of time and the user visits every neighborhood in the city; then a heatmap could expose the silent zones. This is prevented by periodically changing observer IDs.

Random MAC. The Bluetooth specification defines private addresses [17] as resolvable or non-resolvable. Techu is designed to be coupled with non-resolvable private addresses, where all the address bits are random and change based on a timer. Simply relying on these private addresses would break the functionality of a crowd GPS system, as the tag would keep changing addresses after being lost, and the owner would not be able to query for it; we solve this by using a tag attribute to store our ephemeral identifier t_i.

Eddystone. Google's Eddystone framework uses ephemeral IDs to improve privacy [34]; they are time-based and generated by the tag based on a shared secret with the owner. These would allow us to prevent attackers inferring lost devices without using decoy queries, but may significantly increase battery consumption. As part of our future work we will explore how too modify Techu for incorporating these tags without incurring the overhead. Also, since EIDs are time-based and clocks will not be synchronized, the owner may need to query the service for multiple IDs.

BLE 5.0. Early reports [66] for the upcoming BLE 5.0 specification call, among other things, for extending the advertisement packet length. One could argue that this will allow for stronger keys, thus, enabling a design where observers encrypt and store the user's location in the public server, as the concern for forward privacy would be mitigated due to the use of keys of higher sizes. However, this may not be a viable solution, since extensive experimentation [62] has shown that the energy required for packet transmission is *directly* correlated to the payload size; in conjunction with the de-facto high transmission rate of the advertisement packets, this would undermine the tags usability by significantly reducing the battery lifespan.

Business model. As crowd GPS services are becoming increasingly popular, we envision that the low operational costs of Techu could enable the deployment of our system as a community effort by volunteers. Given how such a service could potentially be of use in critical scenarios (e.g., elderly getting lost, children being abducted) and the emergence of relevant services, local Techu instantiations could also be supported by government organizations. Nonetheless, our approach can be readily adopted and deployed as a profitable alternative architecture by tag vendors, as it efficiently addresses the CrowdGps tracker market, while also offering attractive economic benefits by removing the operational and liability costs that stem from safeguarding user-data.

9. RELATED WORK

Crowd GPS. Frank et al. [28] presented a system for locating objects through mobile phones, where other users attempt to locate objects within a predefined defined time window which will result in failure if the object is not within range of users during that time. On the other hand, our design allows owners to obtain the most recent sightings of the device, even if no observer is within range at that moment in time. Furthermore, the heuristic-based selection of potential observers may result in the selection of nodes that have not been in proximity to the lost device, while Techu reports information by *any* member of the crowd.

Recently, Sun et al. proposed SecureFind [70], a system for locating lost objects. One of the design goals of their system is to offer object security, i.e., to prevent the service provider from identifying which objects have actually been lost or infer which users have the device in their vicinity. To achieve that, some participating nodes also return dummy locations as responses to the query. However, this introduces false positives that can mislead the device's owner to wrong locations. Furthermore, as per their threat model of an honest but curious service, they argue that they ensure user privacy as the service only knows the general area of each user (areas are configured to match the granularity that cellular service providers have based on which tower the user is connected

to). However, previous work has demonstrated how users can be deanonymized using location traces of such coarse granularity [24]. Furthermore, their protocol requires proximity of observers to the lost object at the time of the query, while our approach allows the location of objects even in sparsely-populated areas, as we provide information on observations even after the observes leave the object's range, by returning the most recent sightings. Finally, in their system users can issue queries for obtaining location information for tags of other users, effectively being able to track them. While this threat is not part of their threat model, it is a critical threat nonetheless, as their system can effectively be transformed into a malnet, as we demonstrated with the TrackR service.

Cornelius et al. [22] proposed a framework for creating privacy-preserving opportunistic sensing tasks. One of their example test cases was the implementation of the Object Finder service. However, their system has significant privacy drawbacks, as it is vulnerable to user tracking; since the MAC addresses of the objects never change, adversaries can issue queries for the user's device and obtain the user's location. Furthermore, anyone in the crowd can see the requests for specific devices (i.e., everyone knows which devices are lost), and can issue a query for locating those devices.

These systems build upon a centralized service, which requires considerable infrastructure, rendering their deployment as a public service improbable. Thus, users either have to pay for the service, or have their data used for revenue [4]. Techu only needs a public bulletin board as the server, rendering it ideal for a community-driven deployment.

Ho et al. [71] presented a framework that leverages differential privacy for protecting the location privacy of workers during the assignment of tasks in spatial crowdsourcing services. Wang et al. [74] demonstrated how Sybil attacks could impact Waze, a real-time crowdsourced map, by reporting fake incidents such as accidents or congestion.

Device tracking. Ristenpart et al. proposed Adeona [61], a system for locating lost (or stolen) devices. The core of their system is based on the ability of the device to periodically upload reports, i.e., it only works for internet-enabled devices with considerably higher storage capacities, while our system is designed for BLE tags that have limited storage and no internet connectivity, and relies on the *power of the crowd* for locating devices. Furthermore, their system employs round-trip time measurements of known hosts (landmarks) for estimating the geo-location of the device, and potentially requiring communication with the ISP; apart from the impracticality of requiring such communication, these geo-locating techniques [37, 43] achieve very coarse-grained estimations (e.g., within a large metropolitan area) and are, thus, not suitable for this type of functionality. An interesting aspect of their system is that it is built over a Distributed Hash Table, enabling the use of public infrastructure without the need of a trusted central service. As this matches our motivation behind the use of a public server that serves as a bulletin board, we could modify our approach to similarly employ OpenDHT [60] for storing the observation events.

Bluetooth tracking. The issue of location tracking using ubiquitous devices has been extensively studied in the past. Jakobsson and Wetzel [45] discussed, among other attacks, how the location of users could be tracked by an adversary that had deployed or compromised Bluetooth-enabled devices distributed over a (large) area. In this paper, we demonstrated how adversaries can leverage an existing popular crowdsourcing service for deploying such an attack. Following that work a plethora of researchers have explored this issue both for WiFi and Bluetooth devices [39,61,63]. Das et al. [23] showed how the network traffic of BLE wearables could lead to the inference of user activities, or identification of the user within a small group through the user's gait.

Location privacy. A lot of research has focused on location privacy for applications where a user's location needs to be sent to a third party [29, 50]. Many of these studies focus on services that provide location-specific information and, as such, propose solutions that add noise or perturb the user's location [12, 19, 27] or employ cryptographic primitives [57]. These approaches are not suitable for many crowd GPS systems, as accuracy is of prime importance and tags have limited computational power. Also, recent work demonstrated how attackers can accurately pinpoint a user's location in popular services, despite the noise they introduce [58].

10. CONCLUSIONS

As the saying goes, "*there is strength in numbers*". While the crowd's power can be harnessed for tackling significant tasks, with several examples of crowd sourced platforms demonstrating important results, it can also be misused with dire implications. To motivate this discussion, we demonstrated how a very popular service could be trivially misused as a distributed surveillance system, effectively tracking the whereabouts of any user within the system. To proactively prevent such incidents, we designed Techu to offer strong security guarantees, and ensure the location privacy and anonymity of participants from other users, as well as the service itself. In the security analysis of our system we outlined how users are protected against a wide range of attacks, and discussed the tradeoff between privacy and performance overhead. Furthermore, our experiments showed that our system has negligible impact on the user's device in terms of computational overhead and power consumption. Thus, the low operational costs and privacy-preserving nature of Techu can incentivize vendors to adopt our system design. Nonetheless, if crowdsourced GPS services continue to gain traction for critical tasks (e.g., locating missing elderly or children), one could envision the deployment of Techu by volunteers or as a government-led operation.

Acknowledgments

We want to thank the reviewers for their valuable feedback. Special thanks to our shepherd Morley Mao for all her help. This work is based upon research supported in part by the U.S. Office of Naval Research under award number N00014-16-1-2261.

11. REFERENCES

[1] Cryptographic key length recommendation. https://www.keylength.com/en/.

[2] Daily mail - samsonite set to install tracking beacons. http://www.dailymail.co.uk/sciencetech/article-3540967/Now-lost-luggage-tell-Samsonite-set-install/tracking-beacons-new-cases-using-smartphone-app.html.

[3] Daily news - lost items cost americans $5,591: survey. http://www.nydailynews.com/news/national/lost-items-cost-americans-5-591-survey-article-1.2237244.

[4] Forbes - if you're not paying for it, you become the product. http://www.forbes.com/sites/marketshare/2012/03/05/if-youre-not-paying-for-it-you-become-the-product/.

[5] Identity theft resource center - 2015 data breaches. http://www.idtheftcenter.org/ITRC-Surveys-Studies/2015databreaches.html.

[6] Network world - biggest data breaches of 2015. http://www.networkworld.com/article/3011103/security/biggest-data-breaches-of-2015.html.

[7] Reuters - amazon bolsters voice based-platform alexa with investment in trackr. http://www.reuters.com/article/us-amazon-com-alexa-trackr-idUSKCN0XT1GB.

[8] Trackr - pets. http://support.thetrackr.com/hc/en-us/articles/210902166-Can-I-Use-TrackR-On-My-Pet-.

[9] https://www.indiegogo.com/projects/trackr-bravo-the-thinnest-tracking-device-ever--2#/.

[10] Washington post - nsa tracking cellphone locations worldwide, snowden documents show. https://www.washingtonpost.com/world/national-security/nsa-tracking-cellphone-locations-worldwide-snowden-documents-show/2013/12/04/5492873a-5cf2-11e3-bc56-c6ca94801fac_story.html.

[11] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2), 2013.

[12] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914. ACM, 2013.

[13] Associated Press. NYC transit hubs handle flood of lost items. Crain's New York Business, Dec 2013. http://www.crainsnewyork.com/article/20131227/TRANSPORTATION/131229942/nyc-transit-hubs-handle-flood-of-lost-items.

[14] M. Balakrishnan, I. Mohomed, and V. Ramasubramanian. Where's that phone?: Geolocating ip addresses on 3g networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 293–300. ACM, 2009.

[15] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa. Depsky: Dependable and secure storage in a cloud-of-clouds. *ACM Transactions on Storage (TOS)*, 9(4):12, 2013.

[16] K. Bishop. Growth business: GPS tracking the elderly. CNBC. http://www.cnbc.com/2014/03/11/growth-business-gps-tracking-the-elderly.html.

[17] Bluetooth SIG. Specification of the Bluetooth system, 2010. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737.

[18] Bluetooth Special Interest Group. *Bluetooth Specification*, 4.2 edition, 2014.

[19] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Optimal geo-indistinguishable mechanisms for location privacy. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 251–262. ACM, 2014.

[20] V. Cheval, S. Delaune, and M. Ryan. Tests for establishing security properties. In *International Symposium on Trustworthy Global Computing*, pages 82–96. Springer, 2014.

[21] S. Cohen. Locus pocus can track all your bluetooth devices – and other people's, too. VentureBeat, 2015. http://venturebeat.com/2015/06/06/locus-pocus-can-track-all-your-bluetooth-devices-and-other-peoples-too/.

[22] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: Privacy-aware people-centric sensing. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 211–224. ACM, 2008.

[23] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 99–104. ACM, 2016.

[24] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Nature Scientific Reports*, 3, 2013.

[25] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[26] M. Faulkner, M. Olson, R. Chandy, J. Krause, K. M. Chandy, and A. Krause. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *Information Processing in Sensor Networks (IPSN)*, 2011.

[27] K. Fawaz and K. G. Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 239–250. ACM, 2014.

[28] C. Frank, P. Bolliger, C. Roduner, and W. Kellerer. Objects calling home: Locating objects using mobile phones. *Pervasive computing*, pages 351–368, 2007.

[29] G. Ghinita. Privacy for location-based services. *Synthesis Lectures on Information Security, Privacy, and Trust*, 4(1), 2013.

[30] P. Golle and K. Partridge. On the anonymity of home/work location pairs. *Pervasive Computing, Springer*, pages 390–397, 2009.

[31] C. Gomez, J. Oller, and J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, 12(9):11734–11753, 2012.

[32] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196), 2008.

[33] Google. Google cloud messaging. https://developers.google.com/cloud-messaging/.

[34] Google. Growing Eddystone with ephemeral identifiers: A privacy aware & secure open beacon format, 2016. https://developers.googleblog.com/2016/04/growing-eddystone-with-ephemeral-identifiers.html.

[35] M. Green. A riddle wrapped in a curve. https://blog.cryptographyengineering.com/2015/10/22/a-riddle-wrapped-in-curve/.

[36] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[37] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-based geolocation of internet hosts. *IEEE/ACM Transactions on Networking*, 14(6):1219–1232, 2006.

[38] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou. Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm. *ACM Comput. Surv.*, 48(1), 2015.

[39] M. Haase, M. Handy, et al. Bluetrack–imperceptible tracking of bluetooth devices. In *Ubicomp Poster Proceedings*. Citeseer, 2004.

[40] K. Han, E. A. Graham, D. Vassallo, and D. Estrin. Enhancing motivation in a mobile participatory sensing project through gaming. In *Privacy, Security, Risk and Trust (PASSAT) and*

2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, pages 1443–1448. IEEE, 2011.

[41] https://www.thetileapp.com/. Tilebeacon.

[42] https://www.thetrackr.com/. Trackr.

[43] Z. Hu, J. Heidemann, and Y. Pradkin. Towards geolocation of millions of ip addresses. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 123–130. ACM, 2012.

[44] N. Husted and S. Myers. Mobile location tracking in metro areas: Malnets and others. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 85–96. ACM, 2010.

[45] M. Jakobsson and S. Wetzel. Security weaknesses in bluetooth. In *Cryptographers' Track at the RSA Conference*, pages 176–191. Springer, 2001.

[46] C. Jones. 10 wearable safety and GPS devices for kids. The SafeWise Report, 2015. http://www.safewise.com/blog/10-wearable-safety-gps-devices-kids/.

[47] P. Kindt, D. Yunge, R. Diemer, and S. Chakraborty. Precise energy modeling for the bluetooth low energy protocol. *arXiv preprint arXiv:1403.2919*, 2014.

[48] N. Koblitz, Alfred, and J. Menezes. A riddle wrapped in an enigma. Security and Privacy, 2015.

[49] J. Krumm. Inference attacks on location tracks. *Pervasive computing*, pages 127–143, 2007.

[50] J. Krumm. A survey of computational location privacy. *Personal Ubiquitous Comput.*, 13(6), 2009.

[51] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell. Urban sensing systems: opportunistic or participatory? In *Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 11–16. ACM, 2008.

[52] R. Lawler. Land Rover puts Tile's stuff-finding Bluetooth tech in an SUV. engadget, 2016. https://www.engadget.com/2016/04/26/land-rover-puts-tiles-stuff-finding-bluetooth-tech-in-an-suv/.

[53] Maxell. Cr1616 battery datasheet. http://www.maxell.com.tw/images/uploads/2014/10/CR1616_DataSheet_e.pdf.

[54] K. Minami and N. Borisov. Protecting location privacy against inference attacks. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, pages 123–126. ACM, 2010.

[55] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *ACM Sensys '08*, 2008.

[56] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 344–353. ACM, 2013.

[57] S. Papadopoulos, S. Bakiras, and D. Papadias. Nearest neighbor search with strong location privacy. *Proc. VLDB Endow.*, 3(1-2), 2010.

[58] I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis. Where's Wally? Precise user discovery attacks in location proximity services. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 817–828. ACM, 2015.

[59] I. Polakis, S. Volanis, E. Athanasopoulos, and E. P. Markatos. The man who *was* there: validating check-ins in location-based services. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 19–28. ACM, 2013.

[60] S. C. Rhea. *Opendht: A Public Dht Service*. PhD thesis, 2005.

[61] T. Ristenpart, G. Maganis, A. Krishnamurthy, and T. Kohno. Privacy-preserving location tracking of lost or stolen devices: Cryptographic techniques and replacing trusted third parties with dhts. In *Usenix Security Symposium*, pages 275–290, 2008.

[62] J. L. Sandeep Kamath. Measuring bluetooth® low energy power consumption, application note an092.

[63] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, T. Kohno, et al. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Usenix Security*, volume 3, 2007.

[64] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[65] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec. Protecting location privacy: Optimal strategy against localization attacks. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 617–627. ACM, 2012.

[66] B. S. I. G. (SIG). Bluetooth®5 quadruples range, doubles speed, increases data broadcasting capacity by 800%. PRESS RELEASE, 2016. https://www.bluetooth.com/news/pressreleases/2016/06/16/-bluetooth5-quadruples-rangedoubles-speedincreases-data-broadcasting-capacity-by-800.

[67] S. W. Smith. Gagged, sealed & delivered: Reforming ecpa's secret docket. *Harvard Law & Policy Review*, 6, 2012.

[68] M. W. Storer, K. Greenan, and E. L. Miller. Long-term threats to secure archives. In *Proceedings of the Second ACM Workshop on Storage Security and Survivability*, StorageSS '06.

[69] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti. Potshards—a secure, recoverable, long-term archival storage system. *ACM Transactions on Storage*, 5(2), 2009.

[70] J. Sun, R. Zhang, X. Jin, and Y. Zhang. Securefind: Secure and privacy-preserving object finding via mobile crowdsourcing. *IEEE Transactions on Wireless Communications*, 15(3):1716–1728, 2016.

[71] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *Proc. VLDB Endow.*, 7, 2014.

[72] S. Triukose, S. Ardon, A. Mahanti, and A. Seth. Geolocating ip addresses in cellular data networks. In *International Conference on Passive and Active Network Measurement*, pages 158–167. Springer, 2012.

[73] Unacast. Beacons on track to hit 400M deployed by 2020 reports Unacast. BusinessWire. http://www.businesswire.com/news/home/20160126005779/en/Beacons-Track-Hit-400M-Deployed-2020-Reports.

[74] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao. Defending against sybil devices in crowdsourced mapping services. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, pages 179–191. ACM, 2016.