CS262A Final Report: Privacy Preserving Data Muling

Tess Despres
tdespres@berkeley.edu
University of California, Berkeley

Shishir G. Patil
shishirpatil@berkeley.edu
University of California, Berkeley

Alvin Tan
alverino@berkeley.edu
University of California, Berkeley

Jean-Luc Watson
jlw@berkeley.edu
University of California, Berkeley

Abstract

We explore the challenges of implementing a privacy-preserving opportunistic network for Internet of Things (IoT) sensor data collection. Opportunistic networks require no fixed infrastructure and allow edge devices to piggy-back messages through "mule" gateway devices, which can be stationary or mobile. While research interest in such networks has waxed and waned over the years, several large-scale commercial deployments have recently been launched, most notably Amazon's Sidewalk, fueling renewed interest. As these networks become more prevalent, maintaining the privacy of the individuals who participate in them as mules will become increasingly important. In this project, we demonstrate that current implementations of opportunistic backhaul networks leak access patterns through communication metadata, which network providers can leverage to reconstruct location traces. We further argue that opportunistic network and privacy are not mutually exclusive, and suggest some potential directions to strengthen the networks' privacy properties. Last, we build and evaluate a system that applies the Express anonymous communication system (USENIX Security '21) in a data muling context to hide identifying metadata, and evaluate its performance compared to a plaintext baseline backhaul system. Given that opportunistic networks are now seeing large-scale commercial deployments, this project serves as a motivation for designing these systems from the ground up to be privacy preserving.

1 Introduction

Researchers have been working toward the ubiquitous deployment of mobile devices for decades [41], but applications have historically been limited by the need for a corresponding wide-area backhaul infrastructure. Opportunistic mesh protocols [13, 34, 39] have sought to expand connectivity past the range of Internet-connected base stations by creating a mesh network among device nodes. Unfortunately, this still requires providing a set of base stations and assumes that deployed devices encounter each other relatively frequently. On the other hand, commercial efforts over the last few years have converted existing embedded devices, third-party platforms (e.g. mobile phones), and other already-deployed infrastructure into gateways that can vastly expand the reach of a short-range network without relying on a mesh-like structure. In these setups, gateways are pre-existing, often owned by third-parties, and may be mobile, so their locations cannot be planned with pin-point accuracy. By expanding the number and scope of available gateways in this way, coverage is present at such a wide scale that edge devices are likely to pass within connection range at frequent intervals. This enables practical applications – asset tracking [1, 38], urban sensing [5], health monitoring [21], or wildfire tracking [9] – without additional infrastructure deployment cost.

Until recently, these opportunistic backhaul networks have remained vertically-integrated. Apple's *Find My* [1] and the Tile [38] location-based systems, for example, are operated by single entities relying on first-party gateways. In Apple's case, their phone and computer hardware listens for lost devices and relays their location back the owner, while Tile tags advertise their location over Bluetooth Low Energy (BLE) to mobile devices with the Tile app installed. These systems highlight a major issue: any new large-scale deployment has to source their own backhaul mechanism, by either placing enough devices to ensure coverage or by incentivizing consumers to host a gateway application. As a result, commercial efforts are increasingly focused on providing *opportunistic backhaul as a service* for third-party deployments. Google's Physical Web connected Android phones to nearby devices until it shutdown in 2018 due to high amounts of spam [32], Comcast routers broadcast WiFi hotspots that nearby subscribed users can connect to [7], the Helium network incentivizes users to deploy LoRa gateways [20], and Amazon recently launched Sidewalk [2], an opt-out BLE network that provides backhaul through residential Internet connections.

Unfortunately, coalescing responsibility for backhaul routing into a small number of entities (e.g. Apple or Amazon) has significant privacy implications for system participants. Although individual shopping centers and stores have used phone WiFi connections to track customers during their visit [19], widely deployed networks like Sidewalk, with gateway devices at every corner, have the potential to silently track individuals throughout their daily routine with almost no interruption.

We demonstrate that the natural approach to implementing a centralized backhaul system (like in Amazon's Sidewalk) yields an undesirable privacy outcome. In particular, their solution to practical deployment issues like spam-prevention, device authentication, or bi-directional communication yield detailed metadata, including device and gateway identifiers. This allows the backhaul provider to follow participants even when they cannot directly inspect payload contents. Since we expect existing mobile devices, already very capable cellular, WiFi, and BLE-equipped platforms, to be folded into backhaul deployments, the impact of metadata accumulation on individual privacy will only increase over time.

The notion that location-based information can be used to reconstruct a large amount of personal information is not new. Given access to a user's personal *mobility trace*, their identity, home address, work location, or political views can be easily inferred [12]. This project starts off by answering a natural question: realistically,

how much of your private behavior can be inferred by an opportunistic backhaul system? Due to its timeliness and scale, we focus on the specification for Sidewalk [2] as a representative centralized backhaul network with support for third-party applications, and simulate a deployment over real-world mobility data (Figure 1). We demonstrate that with knowledge of deployed gateway locations, the density envisioned by opportunistic backhaul allows for precise mobility trace reconstruction. Further, if limited to only metadata collected for payload routing, we show that, combined with a small amount of prior information, a network can continue to estimate user movements, albeit with lesser accuracy.

We take the stance that privacy does not need to be sacrificed to realize opportunistic systems. To demonstrate this, we build and evaluate a decentralized, privacy preserving opportunistic backhaul system based on Express [17]. Express is a recent anonymous communication system targeted at whistleblower communication, leveraging distributed point functions (DPFs) to allow parties to write messages to a server without revealing which message they sent, guaranteeing write privacy against an arbitrary number of malicious clients and one malicious server. Express is more efficient than previous, heavier anonymous communication systems like [14], with constant-factor overhead per message sent regardless of the number of users and primary reliance on symmetric encryption. We discuss how Express is an ideal starting point for private data-muling systems in Section 4.

Unfortunately, Express is not the panacea for opportunistic backhaul. In our experiments, we evaluate its performance in the context of a data muling system and find that Express is throughput-limited on large sets of mailboxes ($\tilde{1}$M) corresponding to a large sensor deployment, serving single-digit writes per second while incurring latencies in the tens of seconds. In addition, as the database size increases, so does mule power consumption for each write. We outline a number of promising research directions to enhance the performance of a system like Express for backhaul, which would allow a service provider to route payloads at scale.

To summarize, this project makes the following contributions:

(1) We demonstrate that metadata collection by current opportunistic sensor networks can inadvertently leak mule mobility patterns to network operators.
(2) We adapt and implement an Express-based system for data muling that guarantees write-privacy.
(3) We evaluate the drawbacks of our Express-based system and propose potential solutions to make it amenable to the edge.

Paper Organization The rest of the paper is organized as follows. Section 2 reviews prior work in opportunistic sensing networks and anonymous communication. Section 3 demonstrates the ability of a network service provider to trace mule movements based almost solely on routing metadata. Section 4 describes Express system and its adaptation to the opportunistic networking setting and Section 5 details our implementation. We evaluate each component of our privacy-preserving data muling pipeline in Section 6 and conclude with a discussion of project results in Section 7.

2 Background and Related Work

In this section, we first identify how recent commercial designs to expand edge network connectivity differ from existing academic and application-specific deployments. We then discuss Amazon's Sidewalk deployment, and detail how derived location data could seriously compromise client privacy. Finally, we consider work on privacy-preserving communications.

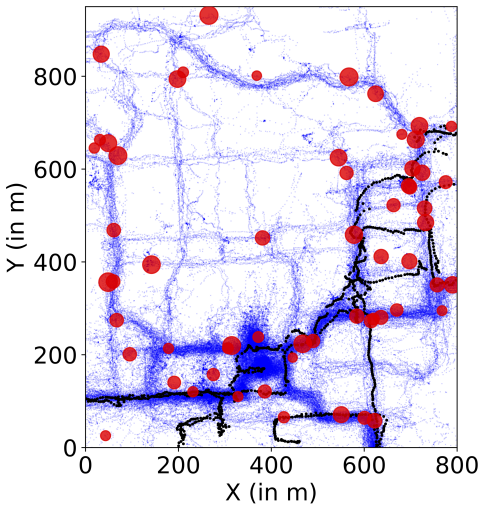

Figure 1: Ground truth device location traces (blue) in the GeoLife dataset [42] and simulated gateway positions (red) on Peking University campus. As a device only connects to a gateway when it enters the gateway's transmission range, a sequence of gateway connections can be used to infer device movement. The black path highlights a specific mobility trace we attempt to reconstruct, discussed in Section 3.

2.1 Opportunistic mesh networks

Much prior work focuses on mobile mesh architectures for data backhaul. Sensors, such as the MULEs (Mobile Ubiquitous LAN Extensions) introduced by Shah et al. [34], transmit packets over short-range wireless links with peers until they reach a base station. The desire to maximize backhaul throughput yielded flooding-based protocols [36, 39], transmitting many message copies. To minimize the resulting power cost of frequent transmissions, probabilistic encounter-based protocols [28, 29, 35] focused on message exchanges to nodes that have a high chance of eventually reaching a gateway. A disadvantage of relying on mesh connections for backhaul is that potential peer nodes are limited to the deployment itself. For example, location-based services such as Tile [38], Apple's *Find My* [1], or Apple/Google's Covid-19 Exposure Notification [10] are heavily vertically-integrated by one central network provider. In the case of *Find My*, Apple devices scan for BLE advertisements to crowd-source an approximate location for a lost device, which is reported back to the device owner in a privacy-preserving manner. Similarly, Tile tags broadcast BLE beacons to phones running the Tile app that then deliver the approximate detected location to the relevant Tile user. These systems are successful in large part due to a years-long deployment effort, through first-party access to millions of mobile phones in Apple's case, or by cultivating millions of active users as Tile has done. However, scaling this approach to support a long tail of mobile

device deployments is not realistic. Consumers are unlikely to continuously run N background applications to support N sensor fleets.

2.2 Backhaul as a service

To provide edge connectivity at scale, the most recent backhaul infrastructure systems have supported third-party deployments by focusing on a gateway-centric design. Just like WiFi or cellular-based networks for more powerful consumer devices, these systems provide gateways to proxy data received over BLE or another low power wireless protocol. Crucially, many such services can be scaled using existing hardware. Google's Physical Web aimed to facilitate interactions with nearby devices, operating on nearly every Android smartphone between June 2016 and its eventual shutdown in December 2018. Adkins et al. [4] demonstrated how web page fetching behavior could be leveraged into an opportunistic backhaul mechanism ferrying small amounts of sensor data. LoRA-based [33] networks enable up to 8 km of communication range, but allow limited bandwidth, require radio line-of-sight, and can be bottlenecked by the quantity of deployed gateways. MachineQ [24] provides a fully-centralized LoRaWAN network, while Helium [20] uses a decentralized ledger to maintain gateways, provide location services, and ensure payment for data backhaul. They incentivize users to deploy their own LoRa hotspots in a density-sensitive manner.

Sidewalk. Recently deployed in June 2021, Amazon's Sidewalk [2] system operates on all Amazon devices (e.g. Amazon Echo, Ring cameras, etc.) and is turned on by default. The network represents a continuation of the trend towards large, centralized gateway deployments, and has significant potential reach, supporting both BLE and 900 MHz (e.g. LoRa) wireless communication. Third-parties can use these gateways to offload data through BLE as they enter the gateway's range, which is relayed to the relevant destination server through a centralized routing service. To deliver application data and enable bi-directional communication, Sidewalk collects a small amount of *routing metadata* at a central network server for each payload. Application-specific data remains end-to-end encrypted throughout the process. Specifically, Sidewalk (1) authenticates the gateway being used and records recently-used gateways for bidirectional communication, (2) collects endpoint identifiers to authenticate devices, (3) keeps gateways time-synchronized to generate correct payload timestamps, and (4) is given the desired server destination for the application data. Unfortunately, while several encryption layers and rotating transmission identifiers protect Sidewalk communication from third-party snoopers, no guarantees can be made on how Amazon itself handles user metadata. The Sidewalk security analysis [2] relies only on a (self-enforced) data retention policy to periodically wipe out routing metadata. In the end, users must place full trust in Sidewalk to deliver on their privacy promises.

2.3 Breaching user privacy with mobility traces

As described above, a system with the broad reach of Sidewalk collects metadata whenever a mobile device comes within range of a particular gateway. These interactions are a function of that device's physical movements, so by coalescing several gateway interactions into a trace, this behavior can be leaked to the backhaul network provider. While methods exist for location obfuscation when queried by an untrusted server [11], devices tracked in this way cannot prevent their position from being reported by an observing gateway. Knowledge of a person's movement patterns represents a substantial breach of privacy. De Montjoye et al. [15] showed that a majority of mobility traces with very low cardinality, containing as little as 4 datapoints, could be uniquely tied to a particular person. These can then be combined with external information (e.g. estimated home and work locations from public records) to deanonymize the trace owner. Srivatsa and Hicks [37] demonstrated this process, using a social network graph to unmask users based on how often their mobility traces intercepted each other. Even indirect location sharing, providing only a relative distance to another party, has been used to recover mobility traces [27]. Finally, methods to combine different location datasets – derived from credit card transactions and ad network data, for example – have been developed to cross-reference individual behavior and derive identities with high accuracy [18, 31]. Once identified, the lack of location privacy leaks a wide array of sensitive information based on visited locations: home addresses [22], political leanings from attending campaign rallies, medical procedures based on visited clinics, or job searches requiring interviews at competing firms [12].

2.4 Anonymous Communication

Given that hiding access patterns is critical, we now look towards efforts in developing metadata-hiding schemes in a stronger threat model. A number of proposed systems use cryptographic techniques to achieve very strong security guarantees.

Riposte [14] proposed a hybrid three-server architecture for anonymous broadcast, where messages are publically revealed for every user in an anonymity set but individual messages cannot be traced to their source. The system makes use of several common tools: a Dining Cryptographer (DC) network where a small set of servers collaboratively compose the eventual public result, a reverse Private Information Retrieval (PIR) scheme for each cleint to write their message without leaking its position, and a multi-party comptuation (MPC) based approach to detecting malicious clients. Similarly, Blinder [3] uses DC-nets for anonymous broadcast, addressing censorship resistance and robustness in the face of an attack. Pung [8] is secure against a malicious global adversary and colluding clients, using PIR once again to build an untrusted key-value store, and Riffle [26] replaces the DC-net with a mixnet to securely shuffle client messages in a multi-server architecture to hide their source. PIR is used to read privately from the shuffled list. Likewise, McMix [6] ditches the PIR and mixing in favor of a pair of MPC protocols.

However, many of these systems have serious practical usability drawbacks. Riposte requires that client communication patterns are data-independent, so every client writes a message to the global billboard every epoch. Similarly, Pung handles reads and writes in bounded rounds, but allows parties to retrieve multiple values at once, albeit with a probabilistic result. MPC-based protocols require significantly more work on the server-side than the equivalent plaintext setting, and in dial-based systems like McMix, only one sender can communicate with a destination party at a time, requiring others to queue to wait their turn.

Recently, a major focus has been on evolving these systems to yield lower overheads. Express [17] achieves significantly better performance in anonymous communication by relaxing the threat model.

Importantly, rather than forcing all users to communicate in each synchronous round, asynchronous reads/writes are allowed but rely on public entities like newspapers generating cover traffic to increase the anonymity set for users. We discuss Express further in Section 5.

3 Mobility Trace Reconstruction

To illustrate the privacy implications of large scale opportunistic backhaul deployments, we design and evaluate a proof-of-concept mobility trace reconstruction, using simulated routing metadata. We simulate a Sidewalk-like deployment where the device ID, gateway ID, and transmission time is collected for each connection. We demonstrate that given the locations of only a fraction of the gateways, a network provider can use timestamped connection sequences to estimate the locations of other gateways in the network and subsequently reconstruct a user's mobility trace by using their connections with various gateways, albeit with nontrivial average error.

3.1 Setup

We use publicly available mobility data from [42] to simulate pedestrians carrying network-enabled endpoint devices while moving around the campus of Peking University in Beijing, China over a 800 by 950 meter area. We simulate a random deployment of opportunistic gateways along pedestrian routes, where they encounter passersby carrying mobile endpoints (e.g. Tile tags [38]). In total, we simulate 12 mobile endpoints and 60 stationary gateways, the trajectories and locations of which are shown in Figure 1. While in practice the number and position of gateways will vary, as will the scale of deployment, our intent is to demonstrate mobility reconstruction in a rich environment.

We divide our analysis into gateway triangulation and location reconstruction. In the former, we use the collected metadata and the locations of a few compromised gateways to estimate the other unknown gateway locations. Then in location reconstruction, we predict the movement of an endpoint over time by tracking the gateways that the device opportunistically connects with.

We make a few key design choices in simulating gateways and device behavior. Devices are represented by a GeoLife [42] trajectory in our target area, and move at a velocity matching their recorded GPS coordinates. We distribute stationary gateways across the campus based on mobility density: where more people travel, more gateways are deployed. All devices broadcast twice per second, with simulated gateway ranges uniformly distributed between 10 and 20 meters. If a device broadcasts within range of a gateway, a connection is logged containing the metadata discussed in Section 2.

We assume that the backhaul network provider acts in an *honest but curious* fashion, gathering a persistent history of transmission metadata (device and gateway identities, and transmission time), but does not collude with other transmitting devices in the network or actively prevent application payloads from being routed.

3.2 Gateway Triangulation

In this section, we detail a method that would allow an adversarial network provider to estimate the locations of all mules participating in the network using only sparse location information. This is a more realistic scenario, as network providers have the flexibility to deploy a few gateways at known locations with high traffic flow in order to perform the analysis.

In particular, by pairing the known locations of a few gateways with the connection sequences generated by devices moving through the area, we can estimate the positions of other nearby gateways through triangulation. Thus, the network provider could derive an estimated position for the other gateways participating in the system, and subsequently use those locations to generate personal mobility traces for each mule.

3.2.1 Estimating pairwise distances For each of the devices in our sample trace, p_i for $i \in \{0,...,11\}$, we have a sequence of connections with the gateways g_k and the times they occurred: $(g_k,t_k)_i$ for $g_k \in \{0,...,59\}$ and $t_k \in [0,T]\}$ for a total trajectory time T. Given this metadata, we can estimate the symmetric matrix $D \in \mathbb{R}^{60\times 60}$ of pairwise distances between gateways in the area. We do this using the time between connections for particular endpoints moving throughout the area. Specifically, for each endpoint p_i, we calculate the list of time differences $(t_k - t_l)$ between connections made with gateways g_k,g_l for connection times t_k and t_l that occurred within two minutes of each other. Since we want an accurate straight-line distance between gateways in order to conduct triangulation, we select the $5th$ percentile value of $(t_k - t_l)$ for each pair of gateways g_k,g_l to use as the time distance estimate, avoiding noise. This gives us a symmetric matrix $T \in \mathbb{R}^{60\times 60}$ of pairwise time differences between area gateways. This matrix tends to be sparse for each endpoint p_i, but including metadata from more endpoints with different paths yields more estimated values in T. We make the simplifying assumption that the endpoint moves at the same speed throughout the entire area, so v is a scalar value, estimated by taking the average of distance between known gateways and dividing it by the time difference between them (i.e. $v = mean(dist(g_k,g_l)/T[g_k,g_l])$ for $g_k,g_l \in known \subseteq \{0,...,59\}$). We can calculate D for speed v from the matrix T using the proportional relationship $D = vT$. The result is our desired pairwise distance matrix D.

3.2.2 Triangulating positions of other gateways Once we have constructed D, our distance estimate between gateway pairs, estimating each location becomes an optimization problem.

Specifically, we solve the following:

$$\min_{pos(g_u)} \sum_{g_k} (||pos(g_u) - pos(g_k)||_2 - D[g_u,g_k])^2$$

where $g_u \in unknown = \{0,...,59\} \backslash known$ represents gateways with unknown locations and $pos(g)$ is the (x,y)-position of gateway g. We minimize the difference between the distances between the predicted positions and the values in D to estimate $pos(g)$ for each gateway. We do this by conducting iterative least squares optimizations on $unknown$ gateways until the positions stabilize. To avoid running into local minima, we instantiate the predicted position values randomly, run 20 predictions with randomized initial positions, and select predictions that minimize the loss. This process gives us gateway position estimates $pos(g_u)$ for $g_u \in unknown$.

The accuracy of this reconstruction depends strongly on the positions of the known gateways and on the amount of mobile endpoint metadata that is available for use. We find that by using 14 known gateway locations, we can reconstruct the positions of the remaining 46 gateways with an average error of 240 meters and median error of 150 meters. While we can estimate some of the gateway

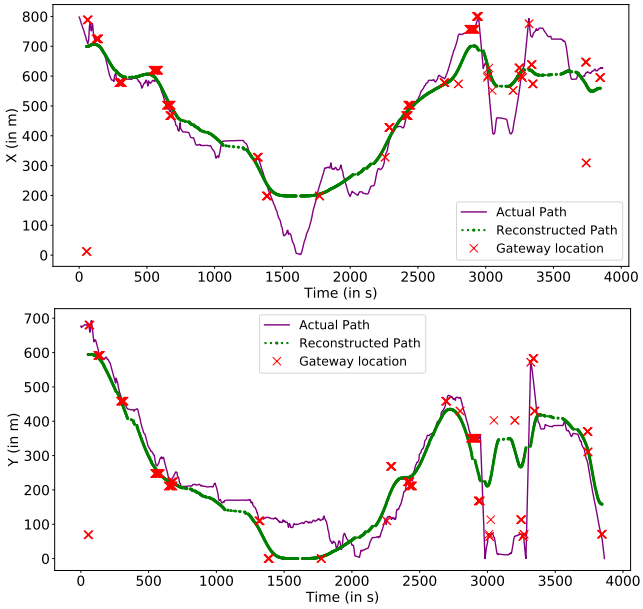

Figure 2: Reconstructed mobility trace estimate, derived solely from connection metadata collected by the backhaul network provider. The purple curves in the top and bottom figures denote the actual position of the sensor along the X and Y axes, respectively, as it connects to various gateways. The green splines represent the estimated mobility trace. Each red X indicates the location of a gateway the sensor encountered.

positions reasonably well, there are many outliers. This is due to the small amount of data that we were working with, which were not amenable to heavy data-mining techniques.

3.3 Location Reconstruction

Using these estimated positions, we can then reconstruct the movement of devices through an area, even if a device never connected to any of the gateways with known positions. We demonstrate this process by focusing on a single mobile device, whose trajectory and visited gateways over the course of an hour are shown as the black overlay trace in Figure 1. We then using linear splines, an interpolation function defined piece-wise by polynomials, between gateway positions to reconstruct the device's movement over time. The reconstructed mobility trace and actual device position over time are shown in Figure 2.

In general, the accuracy of the reconstructed trajectory increases as more connection events are observed. When gateways are sparse, such as between 800 and 2200 seconds into the mobility trace, the spline estimate is oblivious to any detours the device might make. For some of the other endpoints we considered, this sparsity in gateway information caused our spline-based reconstruction to stray up to 400 meters from the ground truth. However, for the relatively well-covered trace in Figure 2, our spline-based reconstruction stays within 105 meters on average from the ground truth device position. While this error is not small, this result still demonstrates the feasibility of such an attack, especially with intelligently-positioned

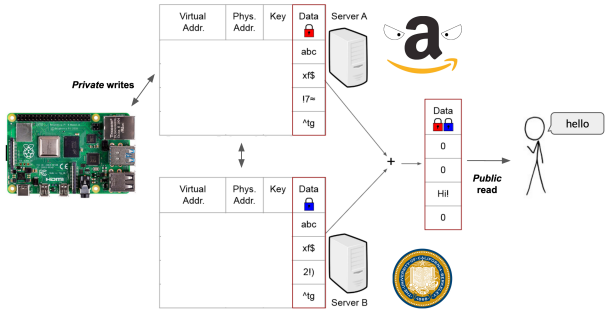

Figure 3: Express based privacy-preserving data-muling system, adapted from Figure 1 of [17]. Mules write privately by sending a payload and DPF that is evaluated on a database table additively secret-shared between two non-colluding servers (e.g. one controlled by Amazon, the other by UC Berkeley). Data consumers can read out of the database by combining the desired row from each server, but this process is public.

gateways and a wealth of connection metadata from progressively larger commercial networks.

4 Hiding Metadata with Anonymous Communication

Given the ability for backhaul service providers to infer personal mobility traces solely from routing data, how can we hide or decouple this information to avoid leakage? One option is to allow mules to write to the server(s) without authenticating them, decoupling which mules (and which sensors) are known to have provided data. Unfortunately, this leaves the entire backhaul system vulnerable to spam, an issue that eventually led to the shutdown of Google's Physical Web in 2018 [32]. Current systems discussed in Section 2.2 that attempt to expand the reach of edge networks (e.g. Sidewalk) require some form of authentication to write data, be it a key provisioned at device manufacturing time, or user credentials like an Apple ID.

Prior work scrutinizing Apple's Find My system has pointed out that using Apple user credentials for both writes and reads to their location database, even with end-to-end encrypted payloads, has the potential to allow for social network reconstruction [23]. They recommend that *read access* be made free of access control, breaking the link between the user of a device that is lost and the device that finds it. Unfortunately, while this would make social network reconstruction difficult, it would not prevent the mobility trace reconstruction from Section 3, as writes of new payloads to the server's database are still fully observable. In addition, while Find My works as an application service that could support free reading, a data backhaul system must charge readers for the use of the system, requiring the network operator to know who the reader is. In this case, the monetization model for network backhaul makes anonymous reads impractical.

Our hypothesis is that integrating an anonymous communication system into the data muling pipeline can support a realistic workload while providing strong privacy guarantees for mules. Specifically, anonymous communication will allow mules to upload payloads to the system's servers and route them to their destinations without revealing important metadata, such as the source of the packet or

who should receive it. At the same time, the system can continue to authenticate both writers and readers to prevent spam, while acting as a sort of "broker" for a database of messages whose contents interested parties may be willing to buy. We primarily consider what information an honest-but-curious backhaul server provider can deduce from mules, and do not focus on cases where the mules or sensors actively collude with the network provider.

To test this, we port a recent system, Express [17], to a data muling environment and compare with a plaintext baseline. Express is described in Figure 3. At its core, it maintains a database of *mailbox* rows whose contents are secret-shared between two, non-colluding servers. Coordination between sensors and their data consumers happens by writing and reading to mailbox addresses known to both parties. Reads are fully public, so interested parties can simply query the desired rows from each server and combine them to retrieve the result. On the other hand, writes are private, where the server knows that a particular mule has performed a write, but not which rows of the database was changed. This is accomplished using a distributed point function (DPF) generated by the client, that returns a secret share of 1 when evaluated on the correct index, and a secret-share of 0 when evaluated on any other mailbox. This indicator can be used to write data into the table without tripping the server to the interested row, but requires an operation be made over every row of the database for every write. In our context, we model data muling with Express by assigning one mailbox to each sensor that the data consumer can poll at regular intervals for new messages.

4.1 Domain-specific requirements

We use Express because several of its design choices dovetail particularly well with data muling.

Public reads. This allows the network operator to note each time a consumer reads from the database, and can use this information to charge for the service accordingly. Compared to a non-private system like Sidewalk, the backhaul operator loses any form of useful metadata: while they may observe what mules are actively writing into the system, no timing information for the sensor packet itself or the sensor or destination IDs for a payload are leaked, since Express hides which mailbox the data is routed to by the client. As a result, the position trace reconstruction from Section 3 is no longer possible.

Existing two party setting. The two-server setting required by Express fits the reality of a data muling setup, as mobile operating systems (the most likely target for a widely-deployed set of mules) make it very difficult to run background sensing applications [25]. As a result, parties like Apple or Google will likely be involved in supporting background muling (similar to Exposure Notifications [10]), while the backhaul broker managing reads and charging for application access can act as the second, non-colluding party.

Lightweight mule cryptography. We gravitated towards Express because it requires primarily symmetric key operations, lightweight auditing protocol, and has a simple interface for contacting each server (encapsulating packets and communicating with just one endpoint), each of which makes it ideal to implement on mule devices. We show that Express can run on a Raspberry Pi in Section 6.4 and evaluate power usage.

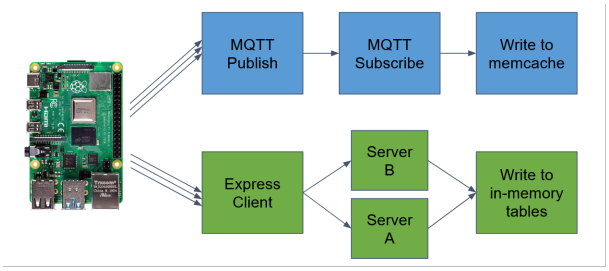

Figure 4: System set-up for evaluation. Here we benchmark mule writes to a plain-text MQTT and privacy preserving Express-based system, with both systems saving received payloads to an in-memory database. The number of mules, sensor density, and other parameters remain consistent.

4.2 Inefficiencies at scale

On the other hand, Express is not entirely suited for this setting (having originally been targeted at whistleblowing). This is primarily due to its use of cover traffic and the low write throughput for large database sizes caused by DPF evaluation.

Cover traffic. Dummy writes in the original instance of Express were meant to increase the anonymity set for users by embedding a script into popular websites to hide when a user was really sending critical data. In data muling, the timing of when we would like to send a packet leaks information, as it most likely indicates that we travelled past a sensor. As a result, we have each mule generate their own set of cover traffic by batching sets of sensor readings and transmitting on a public schedule. This occurs regardless of whether the mule has observed the number of packets to completely fill the buffer. However, it is relatively straightforward to predict that this behavior will result in the server-side component receiving far too many writes in total, decreasing the frequency of useful writing, which we explore experimentally below.

Linear server operations. The need to apply a DPF to every row of the database means that throughput is very slow for large database sizes. This limitation can be seen in [17], where system throughput degrades to single writes per second as the number of rows reaches 10 Millon, so we would expect that as our implementation scales to more sensors, Express's ability to deal with the workload will degrade. We discuss some approaches and next steps to solve this in Section 7.

Next, we describe our implementation of the data muling system and evaluate its performance compared to a plaintext system.

5 System Set-up

To evaluate a privacy-preserving opportunistic network, we set out to build a system that employs Express. Our system consists of three parts – a sensor, a data-mule, and a server to aggregate all of our data.

Sensors: Our sensors are implemented on an nrf52840 platform. The nrf52840 is an ARM Cortex M4f class low-powered microcontroller with 1 MB Flash and 256 KB RAM. The ARM Cortex M4f has been the workhorse of most sensor systems deployed in the real-world [16, 40] and this motivated us to use it as the processor for our sensor system. The actual sensing of the physical phenomenon itself (e.g., air quality sensor, temperature sensor, etc) is simulated without loss of generality. The sensors broadcast data through BLE with a

range of 10 to 20 meters and advertising interval of 2 seconds. It takes 1.5 seconds to establish a BLE connection with a nearby gateway, and data is transmitted across the connection at a rate of 1 Mbps until the connected gateway leaves the range of the sensor.

Gateways: The advertisements from the sensors are picked up by mobile gateways that come within the BLE advertising range of the sensor. We would like to point out to the reader, that this is opposite to the Amazon's Sidewalk structure used in our mobility trace reconstruction, where the sensors are mobile, and the gateways are stationary. We set-up our gateways to be Raspberry-Pi 4s. This is because Raspberry Pi's are powered by Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz with up to 8GB of RAM. We chose Raspberry Pi's as they most closely resemble a mobile phone - in terms of compute and memory capabilities - which we think are the most common data-mules. Once the Raspberry Pi's get the data from the sensor we implement two parallel streams to relay this data to the server - a plain-text baseline implementation and a privacy-preserving implementation, as shown in Figure 4. In the baseline implementation each gateway publishes it's data to a MQTT topic. The subscribes to the topic, and then logs the data to a Memcached service running locally. We use a Memcache in-memory store as opposed to a relational DB to be consistent with Express - which also uses an in-memory store. For the privacy-preserving part, each gateway runs an Express client. The client encrypts the data and posts it to two servers A and B at a predetermined address as described in Section 4.

Server: The servers that aggregate all the data from the mules are AWS ec-2 instances. To have parity in our evaluation, both the MQTT server for the plain-text setting, and the Express servers for the privacy-preseving system are powered by the same configuration with constant network bandwidth (10 Gbps).

6 Evaluation

We evaluate our system to answer the following question: What is the penalty paid for using a privacy preserving system as opposed to a plaintext communication system? We use a simulation, system stress tests, and energy profiling to quantify our system.

6.1 Simulation

To inform parameter selection for our opportunistic network, while avoiding mass sensors deployment, we developed a simulation to evaluate system scalability. Our simulation uses mobility patterns generated from the GeoLife dataset [42]. Sensors in the simulation are scattered around pedestrian routes randomly and generate data with a 10 minute sampling period. This mimics a real-world deployment of, for example, a set of air pollution sensors like those used by PurpleAir [30]. Mules move throughout the city simulation based on mobility patterns. When mules pass within BLE range of sensors, they form BLE connections and pick up sensor packets.

Naively, in the non-privacy preserving case, mules immediately upload data packets after recieving data from a sensor node. Since immediately uploading packets leaks information about how recently a mule passed a sensor, in the privacy preserving case we batch data transfers at a set frequency. While batching hides when data was gathered, this is not completely sufficient. We also employ cover traffic in the form of padding to send uniform sized packets (1024 Bytes)

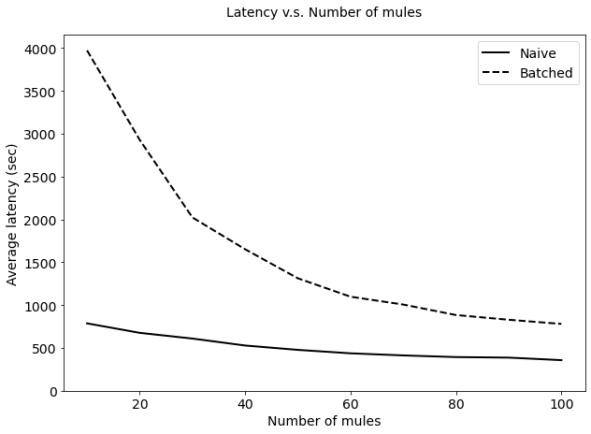

Figure 5: Data transfer latency as the number of active mules increases. As more mules roam around the simulated area, a sensor is more likely to see and transmit to a nearby mule.

at the batch frequency. In this way, the server is only privy to uniform packet transfers at set frequencies, which leaks no information about where the mule has traveled. Unfortunately, we can not get this guarantee for free, there is significant overhead which we evaluate for latency, throughput, and goodput. We define goodput as the ratio of actual sensor data to total data, where total data includes padding.

We find that latency decreases as the number of active mules increases for both the batched and naive data transfers Figure 5. However, the batched upload schedule has a significantly higher latency. This is because the mules wait until a scheduled send to dispatch data. We also find that the throughput, or amount of data the system must transfer, increases with the number of active mules Figure 6. This becomes a problem in the batched and padded case, because the throughput increases linearly. Additionally, as the number of mules increases, the amount of goodput decreases, meaning there is more wasted overhead Figure 6. In a real world deployment, this would quickly become untenable as the number of active mules increases. We discuss potential solutions to this problem in the Section 7.

6.2 System

In addition to simulating the data transfer timing between the sensor and mule, we also evaluate the upload latency and throughput from the mule to a server using our Express [17] implementation compared with a plaintext publish subscribe system. As a baseline, we developed an MQTT publish subscribe pipeline which stores data locally in a Memcached service. We use an AWS EC2 Linux machine with 96 CPUs to write in parallel to our two system pipelines. Each CPU on the machine acts as a separate mule device.

As the number of mules simultaneously writing to our system increases, the plaintext latency stays approximately constant while the Express latency increases Figure 7. For instances of Express with more rows, the latency increases more abruptly. This matches our hypothesis that Express will struggle under a large parallel load. Additionally, we see a similar trend with throughput. As the number of mules sending in parallel increases, throughput scales linearly

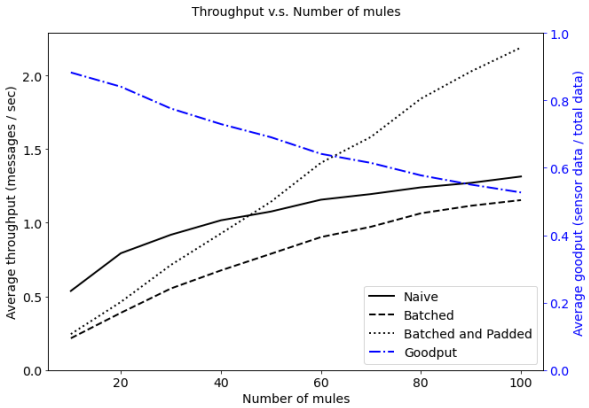

Figure 6: Data transfer throughput as the number of active mules increases. Throughput increases as more mules join, but due to their batched upload schedule, the amount of padding increases linearly and results in worse goodput.

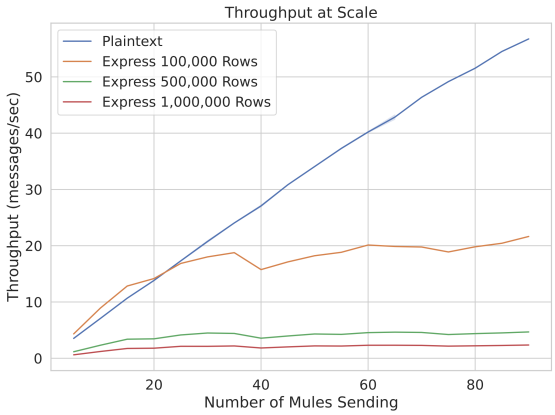

Figure 8: Throughput increases linearly as more mules write to the baseline plaintext data broker. Express's throughput depends on the number of rows in the database and caps out very quickly, supporting a low number of writes/second.

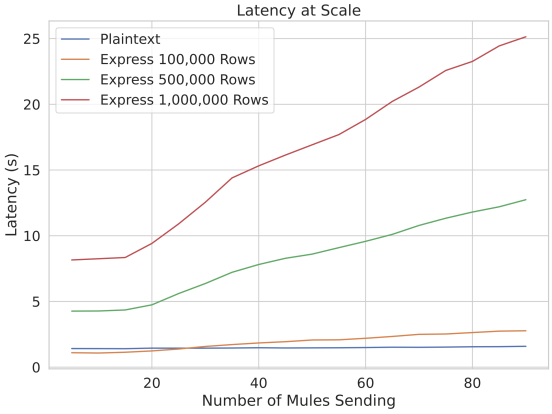

Figure 7: The per-write latency increases as more mules attempt to write to the Express deployment in parallel. A larger number of mailbox rows yields significantly slower write times.

for the plaintext deployment, but Express is quickly limited to low throughput (Figure 8).

6.3 End-to-end buffered writes

While we can separately evaluate the local latency of data muling pickup by simulation and the nominal throughput of our Express-based servers separately, we would also like to understand how well the system handles an end-to-end workload. We evaluate this using the batch upload schedule of each simulated mule to queue writes to the backhaul servers. If the end-to-end deployment can support the incoming sensor traffic, the number of pending writes queued on each mule should remain low over a long duration. We ran the simulation forward for 30 minutes, measuring the average queue of pending writes on each mule for three Express deployments of ten thousand, one hundred thousand, and one million rows. Each mule seeks to complete a batch of one hundred writes every 60 to 600 seconds, randomly chosen at each interval.

Our results, shown in Figure 9, indicate that the throughput limits seen in Figure 8 are critical to ensuring that sensor payloads do not accumulate without limit on the mules themselves. As the database size increases from ten thousand to a million rows, the ability of the servers to keep up with mule writes decreases rapidly. Encouragingly, a smaller Express-based instance of ten thousand rows was able to adequately support the required throughput, resulting in buffer sizes hovering around 0 on average, but larger row counts led to constantly increasing queues with no signs of clearing data. Realistically, mules would at some point decide to stop receiving sensor packets until their buffers depleted, discarding any packets they saw after that point.

6.4 Energy

Since the mules in our system are low power devices, such as cell phones, we have to be cognisant of how much power is being consumed by our system. Excessive power draw will drain battery life and reduce the amount of users who are willing to participate in the network. We evaluate the power consumption of an Express client running on a Raspberry Pi device by using a Chroma 66205 Digital Power Meter. Power is recorded every 0.2 seconds for a minute while the Raspberry Pi is transferring data. A data packet is sent every 10 seconds. A 10 second waiting period allows the power to return to an idle state between data transfers. The Raspberry Pi in an idle state pulls approximately 2.6 Watts.

When running the MQTT plaintext publish subscribe system, we observe power peaks to approximately 3.3 Watts. When running Express, we observe power peaks between 3.3 Watts and 3.9 Watts depending on the number of rows. The raw power traces for both plaintext and Express implementations are shown in Figure 10. We find that as the number of rows in Express increases, so to does the

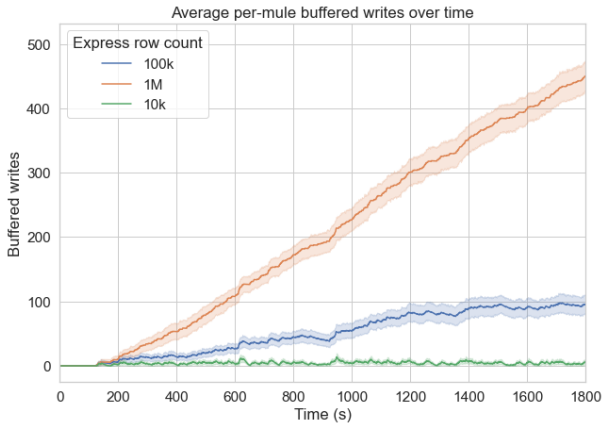

Figure 9: Large Express instances cannot support a simulation-based data muling workload, yielding strictly increasing write queue sizes on each mule. A small instance with ten thousand rows can successfully handle the workload.

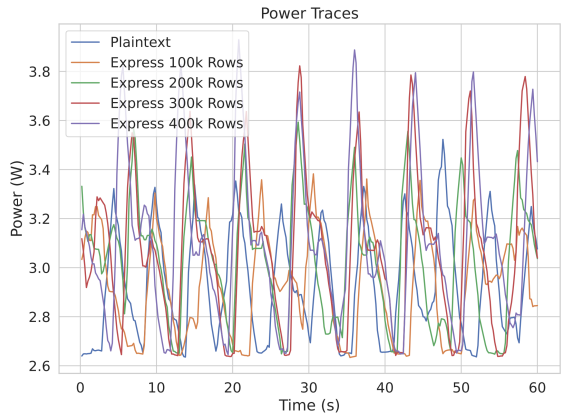

Figure 10: Power traces of Express and plaintext. The power consumption spikes as data is transferred. For larger amounts of Express rows we see higher peak power draw as compared with the plaintext implementation.

power consumption. To put these numbers in perspective, when all 4 CPU cores are stressed on the Raspberry Pi, we observe a power consumption of 6.1W.

While the power increases as function of express rows, it stays at a manageable level up until 400k rows. The average power as a function of Express rows is shown in Figure 11. Unfortunately, we were unable to measure beyond 400k rows because the Raspberry Pi has a finite memory size. Since rows correspond to the number of sensor mailboxes Express can support, we will need to examine ways to scale beyond 400k in the future. A few potential possibilities to accomplish this are introduced in Section 7.

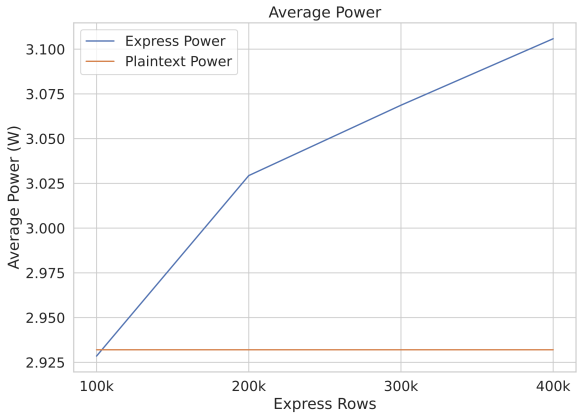

Figure 11: Average Power of Express and plaintext. As the amount of rows in Express increases, the average power over our one minute measurement window also increases. The plaintext average power is shown in orange for reference.

7 Discussion

Recent industrial deployments of opportunistic networks for the long tail of mobile devices indicate that interest in this area will only continue to increase. Previously, opportunistic networks were primarily small-scale academic projects, but now commercial systems (e.g Tile, *Find My*, Sidewalk) have real-world impact. As interest ramps up, and more companies begin to support third-party backhaul, the impact of systems on user privacy should be examined.

We argue that user privacy should be treated as a first-class priority for opportunistic networks. We hope that the proof-of-concept demonstration of location trace reconstruction will spark conversations around the unintended consequences of deploying large-scale opportunistic networks. As a starting point, we implemented an Express-based PIR system and explored the performance of such a system through both hardware measurements and simulation. In this section, we briefly recapitulate what can be achieved with an Express-based system, what challenges we have identified, and finally what next steps should be considered for a commercially viable privacy preserving system.

7.1 Successes

Our project, while preliminary, achieves a few of the properties that are necessary for a real-world privacy preserving opportunistic network.

Accountability. Using Express, we are able to authenticate who is writing into, or reading from, the database. We are also able to track the volume of data being read, which enables the network provider to charge users for the backhaul provided.

Metadata Obfuscation. By obfuscating the mailbox row to which a gateway writes data, we effectively decouple the gateway from the endpoint device which generated it, preventing the system from inferring location data. Through randomly-spaced batched uploads, the mules obfuscate the activity patterns of the mule owners, such that all mules are indistinguishable based on their upload patterns and metadata.

One-Way Communication. As expected from data backhaul infrastructure, the system is capable of collecting data from IoT endpoints and transferring it to the cloud for consumption, albeit in a best-effort manner. For latency and data loss tolerant applications, this is acceptable.

Performance Estimates. Based on our simulations, we are able to identify and explore the tradeoff between latency and goodput as we vary parameters such as mule batch size and sensor sampling frequency. Given an example application (e.g. air quality monitoring) and its matching expected sensor and mule counts, this provides us with estimates of what latency and overhead we could expect. Through our software and hardware implementations, we are also able to explore the throughput of the server component under heavy load and quantify the power overhead of communicating with the Express instance on Raspberry Pi gateways.

7.2 Challenges

However, our system fails in some obvious ways that provide interesting future research directions for this project.

Limited Throughput. The additional overhead of anonymous communication, in this case Express, limits the amount of messages that can be written into the servers at any time. We do see that throughput increases as the size of the database decreases, which means that a large database could be sharded into many smaller tables for higher throughput. However, smaller tables can support less endpoint devices, and if tables get too small (or if endpoints get too dense), the locality of a shard may reveal additional location information on the gateways and sensors. This is a balance that will need to be assessed carefully; using already-known information like a mule's IP address may allow for sharding into smaller databases while revealing no additional information.

Excessive Padding. The goodput values generated in Section 6.1 assume that the gateways are constantly moving and picking up new packets to upload. However, this is not what one would expect if the gateway were a smartphone, for example. A gateway might instead be mobile in the morning and evening during a commute to and from work, with some additional bursts of mobility throughout the day for lunch or exercise. However, we expect that the smartphone mule will be stationary a majority of the time, barely picking up any packets but constantly uploading dummy buffers. Thus, the expected goodput for such a gateway would be substantially worse than what our simulation currently predicts.

Therefore, we need to consider alternatives to batching, such that the amount of cover traffic uploaded to the cloud is limited, while still obfuscating times of activity for each mule. This could potentially be achieved by implementing an upload trickle, where a gateway slowly uploads collected packets throughout a day, but such a protocol would greatly increase the expected latency of the data received, while still exposing some timing information if not careful.

7.3 Next Steps

Even without these issues, our current prototype system lacks properties that are necessary for commercial deployment.

Authentication and Spam Prevention. While gateways can be authenticated as they write into the servers, it is difficult to provide and revoke authentication for endpoint devices sending data to gateways at scale without leaking gateway-sensor pairings. This is made more difficult as gateways may not have internet connectivity during their interactions with endpoint devices.

Since the source of data is obfuscated during the write into an Express server, it is not clear how to distinguish between an endpoint providing useful data that is being consumed and an endpoint that is simply spamming the network with data that is never read from the servers by a data consumer (and thus never paid for).

Additionally, there are no guarantees that the gateways will behave honestly. A malicious gateway could overwrite the data in each of the servers hosting the secret-shared database, preventing data from being collected. Express originally avoided this problem because only the person who generated and sent the data knew the corresponding destination address, so the probability of another malicious party guessing and overwriting the correct address was very low. However, in a data muling environment, the endpoint device needs to give the mailbox address to the gateway for them to upload their data, so a malicious gateway knows exactly which addresses contain useful information.

Bidirectional Communication. Although bidirectional communication is not necessary for some applications, it is useful to have for an endpoint to confirm that data has been successfully received. Most PIR-based systems like Express do not support privacy-preserving bidirectional communication. For one, it is not obvious which gateways see which endpoints (this information is, in fact, purposefully hidden), making it difficult for data to be passed through a gateway to a particular endpoint device. Additionally, there is no guarantee that a gateway that passed by a particular endpoint device would ever come in range of that device again. Thus, an opportunistic network designed to preserve privacy makes bidirectional communication a surprisingly difficult challenge.

8 Conclusion

In this paper, we demonstrate that routing metadata relayed in existing opportunistic backhaul systems can leak personal location information. Our analysis highlights a real privacy risk: with sparse gateway location knowledge, mobility traces can be recreated with a mean error of 105 meters. We expect this error to be less in a dense commercial deployment with data collection occurring over a long period of time. Since such systems are being deployed at scale, privacy and security must be taken into consideration.

With this in mind, we designed and implemented an Express-based privacy-preserving data backhaul network, which can obfuscate any location and timing metadata from gateway connections, at the cost of requiring a more intensive private write process. We also designed a simulation to stress test the system at scale, and found that the current state of Express is generally unsuitable for a data backhaul network, in terms of both performance and functionality, except at a fairly small scale. In particular, the limited throughput for mule writes and our current method of batched uploads to hide when mules see individual sensors add considerable overhead at scale.

We look forward to implementing a more scalable server-side design and modifying the mule upload schedules to yield a more practical system, and hope this project can motivate research into new opportunistic networking rooted in privacy.

References

[1] Find my network. https://developer.apple.com/find-my/. Accessed: 09.24.2021.

[2] Amazon sidewalk privacy and security whitepaper. https://m.media-amazon.com/images/G/01/sidewalk/final_privacy_security_whitepaper.pdf, 2021.

[3] Ittai Abraham, Benny Pinkas, and Avishay Yanai. Blinder: Mpc based scalable and robust anonymous committed broadcast. *IACR Cryptol. ePrint Arch.*, 2020:248, 2020.

[4] Joshua Adkins, Branden Ghena, and Prabal Dutta. Freeloader's guide through the google galaxy. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, pages 111–116, 2019.

[5] Joshua Adkins, Branden Ghena, Neal Jackson, Pat Pannuto, Samuel Rohrer, Bradford Campbell, and Prabal Dutta. The signpost platform for city-scale sensing. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 188–199. IEEE, 2018.

[6] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. Mcmix: Anonymous messaging via secure multiparty computation. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1217–1234, 2017.

[7] Suzan Ali, Tousif Osman, Mohammad Mannan, and Amr Youssef. On privacy risks of public wifi captive portals. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 80–98. Springer, 2019.

[8] Sebastian Angel and Srinath Setty. Unobservable communication over fully untrusted infrastructure. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 551–569, 2016.

[9] Thierry Antoine-Santoni, Jean Francois Santucci, Emmanuelle de Gentili, and Bernadette Costa. Using wireless sensor network for wildfire detection. a discrete event approach of environmental monitoring tool. In *2006 First International Symposium on Environment Identities and Mediterranean Area*, pages 115–120. IEEE, 2006.

[10] Apple/Google. Exposure notification - bluetooth specification. https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf?1, 2020.

[11] Claudio Agostino Ardagna, Marco Cremonini, Ernesto Damiani, S De Capitani Di Vimercati, and Pierangela Samarati. Location privacy protection through obfuscation-based techniques. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 47–60. Springer, 2007.

[12] Andrew Blumberg and Peter Eckersley. On locational privacy, and how to avoid losing it forever. https://www.eff.org/wp/locational-privacy, 2009.

[13] J. Burgess, B. Gallagher, David D. Jensen, and B. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–11, 2006.

[14] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE, 2015.

[15] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3(1):1–5, 2013.

[16] Samuel DeBruin, Branden Ghena, Ye-Sheng Kuo, and Prabal Dutta. Powerblade: A low-profile, true-power, plug-through energy meter. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 17–29, New York, NY, USA, 2015. Association for Computing Machinery.

[17] Saba Eskandarian, Henry Corrigan-Gibbs, Matei Zaharia, and Dan Boneh. Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, pages 1775–1792, 2021.

[18] Jie Feng, Mingyang Zhang, Huandong Wang, Zeyu Yang, Chao Zhang, Yong Li, and Depeng Jin. Dplink: User identity linkage via deep neural network from heterogeneous mobility data. In *The World Wide Web Conference*, pages 459–469, 2019.

[19] Brian Fung. How stores use your phone's wifi to track your shopping habits. *The Washington Post*, 19, 2013.

[20] Amir Haleem, Andrew Allen, Andrew Thompson, Marc Nijdam, and Rahul Garg. Helium: A decentralized wireless network. 2018.

[21] David Hasenfratz, Olga Saukh, Silvan Sturzenegger, Lothar Thiele, et al. Participatory air pollution monitoring using smartphones. *Mobile Sensing*, 1:1–5, 2012.

[22] Wajih Ul Hassan, Saad Hussain, and Adam Bates. Analysis of privacy protections in fitness tracking social networks-or-you can run, but can you hide? In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 497–512, 2018.

[23] Alexander Heinrich, Milan Stute, and Matthias Hollick. Openhaystack: a framework for tracking personal bluetooth devices via apple's massive find my network. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 374–376, 2021.

[24] Jim Hildenbrand. Simplifying wireless iot gateway deployment. https://machineq.com/post/simplifying-wireless-iot-gateway-deployment, 2019.

[25] Noah Klugman, Meghan Clark, Pat Pannuto, and Prabal Dutta. Android resists liberation from its primary use case. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 849–851, 2018.

[26] Albert Hyukjae Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle: An efficient communication system with strong anonymity. 2015.

[27] Huaxin Li, Haojin Zhu, Suguo Du, Xiaohui Liang, and Xuemin Shen. Privacy leakage of location sharing in mobile social networks: Attacks and defense. *IEEE Transactions on Dependable and Secure Computing*, 15(4):646–660, 2016.

[28] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20, 2003.

[29] S. Nelson, Mehedi Bakht, R. Kravets, and A. F. Harris. Encounter-based routing in dtns. *IEEE INFOCOM 2009*, pages 846–854, 2009.

[30] PurpleAir. Purpleair: Real-time air quality monitoring everyone can use. https://www2.purpleair.com/, 2021.

[31] Christopher Riederer, Yunsung Kim, Augustin Chaintreau, Nitish Korula, and Silvio Lattanzi. Linking users across domains with location data: Theory and validation. In *Proceedings of the 25th International Conference on World Wide Web*, pages 707–719, 2016.

[32] Nayak M Ritesh. Discontinuing support for android nearby notifications. https://android-developers.googleblog.com/2018/10/discontinuing-support-for-android.html, 2018.

[33] Semtech. Lora and lorawan. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/, 2021.

[34] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: modeling a three-tier architecture for sparse sensor networks. *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pages 30–41, 2003.

[35] Soamdeep Singha, Biswapati Jana, S. Jana, and N. Mandal. A survey to analyse routing algorithms for opportunistic network. *Procedia Computer Science*, 171:2501–2511, 2020.

[36] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05*, 2005.

[37] Mudhakar Srivatsa and Mike Hicks. Deanonymizing mobility traces: Using social network as a side-channel. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 628–637, 2012.

[38] Tile. How tile works. https://www.thetileapp.com/en-us/how-it-works, 2021.

[39] Amin Vahdat. Epidemic routing for partially-connected ad hoc networks. 2009.

[40] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. Farmbeats: An iot platform for data-driven agriculture. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 515–529, 2017.

[41] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3), 1991.

[42] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW '09*, 2009.