ProLoc: Robust Location Proofs in Hindsight

Roberta De Viti*
Pierfrancesco Ingo*
Isaac Sheff
Peter Druschel
Deepak Garg
Max Planck Institute for Software Systems (MPI-SWS)
Saarland Informatics Campus
Saarbrücken, Germany

ABSTRACT

Many online services rely on self-reported locations of user devices like smartphones. To mitigate harm from falsified self-reported locations, the literature has proposed *location proof services* (LPSs), which provide proof of a device's location by corroborating its self-reported location using short-range radio contacts with either trusted infrastructure or nearby devices that also report their locations.

This paper presents ProLoc, a new LPS that extends prior work in two ways. First, ProLoc relaxes prior work's proofs that a device was *at* a given location to proofs that a device was *within distance d* of a given location. We argue that these weaker proofs, which we call *region proofs*, are important because (i) region proofs can be constructed with few requirements on device reporting behavior as opposed to precise location proofs which cannot, and (ii) a quantitative bound on a device's distance from a known epicenter is useful for many applications. For example, in the context of citizen reporting near an unexpected event (earthquake, fire, violent protest, etc.), knowing the verified distances of the reporting devices from the event's epicenter would be valuable for both ranking the reports by relevance and flagging fake reports.

Second, ProLoc includes a novel mechanism to prevent collusion attacks where a set of attacker-controlled devices corroborate each others' false locations. Ours is the first mechanism that does not need additional infrastructure to handle attacks with made-up (virtual) devices, which an attacker can create in any number at any location without any cost. For this, we rely on a variant of the TrustRank algorithm applied to the self-reported trajectories and encounters of devices. Our specific goal is to prevent *retroactive* attacks where the adversary cannot predict ahead of time which fake location it will want to report, which is the case for the reporting of unexpected events.

1 INTRODUCTION

Nowadays anyone with a smartphone can capture, document, and disseminate news. This collective power to share real-time, unfiltered, accounts of events is the essence of *citizen journalism.* By democratizing the reporting of information, citizen journalism represents a seismic shift from traditional media and has been reshaping the landscape of news reporting, with social media platforms like Instagram, Facebook, and Reddit serving as outlets for news dissemination.

Citizen journalism plays a crucial role in offering pervasive, on-the-ground perspectives, particularly during crises or unexpected events. For instance, the Iranian presidential election protests in 2009 and the Egyptian uprising in 2011 showcased the profound impact of citizen reporting, and its capability to bypass government censorship and mobilize collective action [2]. Similarly, the 2005 London bombings were extensively covered by citizens' on-site photos and videos, which traditional news outlets later incorporated, recognizing the value of the information that citizens provided [4].

However, citizen journalism is not subject to the same journalistic scrutiny as news in mainstream news media. This means citizen reports may be of varying quality, reflect bias, and even present an avenue for deliberate propaganda and misinformation. Therefore, the ability to verify the quality and legitimacy of citizen journalistic reports is crucial. However, doing so presents a complex, multi-layered challenge. We take an important step towards addressing this challenge: validating the location of *geo-tagged* reports.

Verifying the geolocation of a citizen at the time of an event is relevant for both the quality and veracity of the citizen's report about the event. First, a verified geolocation speaks to the quality of the report, because a citizen who was near the event is more likely to have observed the event first-hand. Second, citizen journalism is most relevant for events whose time and location is not predictable (e.g., natural disasters, accidents, attacks) because mainstream media and journalists are not likely to be present initially. For such

---

*These authors contributed equally to this work and are listed alphabetically.

reports, a verified space-time location speaks to the authenticity of the report and any associated digitally captured material (audio, video, still images). This is because an adversary intent on submitting biased or false reports about such an event would have to have a device physically near the (unpredictable) time and location of the event, which is a challenge even for powerful adversaries like troll farms.

Requirements and prior work. Prior work has proposed several *location proof services*[1] or LPSs that provide independent evidence of a device's (self-reported) geolocation [10, 13, 20, 26, 31, 33, 35, 38]. However, citizen journalism and similar scenarios place specific requirements on a LPS. First, the system should be able to generate a location proof for an arbitrary time and location a user visited. Second, the system must be robust to collusion attacks that seek to generate false location proofs using many fictitious devices. Existing LPSs are not a good match for these requirements. Some systems depend on radio contact with trusted WiFi base stations at known locations, which is unlikely to be available at the location of an unpredictable event [20, 26, 31]. Other systems instead depend on short-range radio contacts with nearby devices that can corroborate each others' location reports [33, 35, 38]. However, they require radio contact with nearby devices at the precise time and location in question. If such radio contacts exits, then a proof for the specific space-time location is issued; otherwise, no proof can be issued at all. Lastly, while some existing LPSs include partial defenses against collusion attacks, none are robust against large-scale collusion by fictitious devices.

ProLoc. To this end, we present a new LPS, *ProLoc*. For a given device d, instant in time t, and a required number of *witnesses N*, ProLoc can determine retroactively a *feasible region S* within which d could have been present at time t, given the evidence collected from N witnesses. Witnesses are unique devices that can independently corroborate d's presence around t. ProLoc determines the feasible region by correlating, during a period around t, (1) short-range radio encounters between d and nearby witness devices; (2) the geolocations recorded by d and the witnesses around t; and (3) uses the transportation network map in and around the feasible region to see how far a device could have traveled between radio contacts and geolocation reports.

By weakening (discrete) location proofs to regions proofs, ProLoc is able to utilize short-range radio contacts and geolocations that were recorded *around* but not necessarily *precisely at* time t. Therefore, it is able to produce location proofs in many scenarios where existing systems cannot. On the other hand, we will show that region proofs are useful towards assessing the veracity of citizen journalistic reports.

We note that media platforms that support citizen journalism typically record geolocations periodically reported by users already. ProLoc additionally relies on Bluetooth Low Energy (BLE) advert exchanges, which smartphones can emit and record with high energy efficiency.

LPSs that do not rely on trusted infrastructure are susceptible to collusion attack. In particular, a device can fake radio contacts with colluding devices to be able to obtain a fake location proof. ProLoc focuses on *retroactive attacks*, where the adversary cannot predict upfront for what time and location it wishes to generate a false proof. We consider this assumption consistent with scenarios like citizen journalism, which is most relevant for unexpected events —- otherwise, traditional news websites would have their own journalists present at the event.

An adversary may create a large number of *fictitious devices* that can corroborate false location reports. ProLoc enables defences against a large class of these collusion attacks involving fictitious devices. ProLoc's defence relies on trust computations based on devices' connectivity graph. To the best of our knowledge, ProLoc is the first infrastructure-less LPS robust to large-scale retroactive attacks with fictitious devices.

We believe that ProLoc provides an important step towards enhancing the reliability of citizen journalism reportings. ProLoc offers: (i) a practical method to provide region proofs in hindsight, based on corroborating evidence recorded around the time of an event; and (ii) the first infrastructure-less defence against a large class of easy-to-mount collusion attacks with many fictitious devices, and a limited number of adversarial physical devices.

This paper's contribution includes (i) *ProLoc*, a system that validates location reports retroactively, and produces region proofs based on the device's radio contacts around the time on question; (ii) an analysis of powerful collusion attacks and the first defense that can tolerate large-scale collusion, including multiplicity and stalking, without requiring trusted infrastructure; and, (iii) an experimental evaluation based on a simulated dataset and on the DTU dataset of locations and BLE contacts collected from 850 devices over a period of 3 years. In the remainder of this paper, we provide an overview of ProLoc's design in § 2. Then, we discuss ProLoc's API in §3, which describes how we consolidate user trajectories to generate location proofs, and we present our defense against large-scale collusion attacks in §4. We report an experimental evaluation of ProLoc in §5 and §6, and discuss related work in §7.

2 DESIGN PRELIMINARIES

Location proof vs. location provider. A *location proof service* (LPS) like ProLoc is orthogonal to a *location provider*

---

[1]Location proofs are not proofs in the mathematical sense; we adopt the term because it is used in the literature.

service; the two serve different purposes and may provide different results depending on the scenario. For instance, the location provider of a smartphone carried by a lone hiker in a national park may provide location with a precision of a few meters, which is great for navigation and even to provide coordinates in an emergency call. However, if the hiker provides this location to the third party, then, from the third party's perspective, the only root of trust for the location is the hiker's phone. A location proof service, on the other hand, could provide at best place the hiker somewhere in a large feasible region based on the most recent contacts the phone had at the parking lot and the distance the hiker could have traveled in the interim. Such a proof could still be useful if the hiker wishes to subsequently prove to a third party that they were in or near the park. On the other hand, the smartphone of a user in a busy downtown hotel not only obtains a precise location, but may be able to obtain a location proof with a tight feasible region placing him within tens of meters of a hotel, because of its numerous BLE contacts with a large and diverse set of devices. As a result, the user can prove, for instance, that they actually were close to the hotel when they claim to have witnessed a fire in the hotel.

State-of-the-art. As detailed §7, prior work has shown that devices can obtain location proofs from WiFi APs or from nearby devices via short-range radio [10, 13, 20, 26, 31, 33, 35, 38]. However, existing systems can attest locations only when a location proof was obtained *a priori* from nearby APs or devices at the time, or in the case of [10], when a *trustworthy* device was present at the exact time. Since proof generation is an expensive operation, it is not feasible to continuously generate proofs for all locations visited to anticipate a possible future need. Hence, existing systems cannot be used to generate location proofs *a posteriori* (without advance planning), as would be needed in the context of citizen journalism. In contrast, we designed ProLoc to generate proofs a posteriori.

To mitigate collusion attacks, some prior work has used plausibility and consistency checks on device trajectories, radio contacts, witness selection, and voting record [10, 33, 35, 38]. However, no existing work is robust against collusion attacks involving large numbers of fictitious devices that corroborate each others' statements. ProLoc includes a defense that works against retroactive collusion attacks even with large numbers of fictitious devices.

ProLoc model. Every device participating in ProLoc locally stores and periodically uploads its *device history*, which is a time series of geographical *locations* and received *near-range radio transmissions* from peers. Locations originate from the location service of the device's platform (e.g., iOS or Android), which typically relies on a combination of GPS,

Figure 1: User device trajectories (colored lines) containing location reports, connected by encounters.
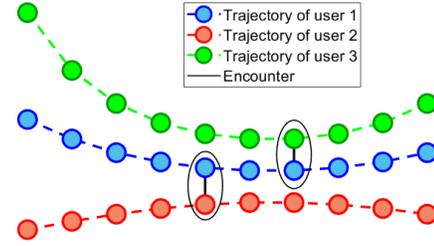
cellular, and WiFi-based methods. Locations are recorded on a regular basis (e.g., every 5 minutes while a device is on).

Near-range transmissions occur over short-range radio like Bluetooth Low Energy (BLE). As explained in prior work [13, 25, 34], near-range transmissions typically take the form of an ephemeral id, a high entropy value unique to the transmitting device and epoch. A device history is a time series of location reports (the device's trajectory) interspersed with ephemeral ids received from nearby devices.

An *encounter* is a period of co-location between two devices, called the encounter *peers*. Individual received ephemeral ids do not directly constitute encounters. An encounter is indicated only when both peers receive each others' transmissions regularly for the duration of the encounter. Detecting encounters requires correlating the histories of potential encounter peers by the backend. Encounters tie histories of different devices to each other, as illustrated in Figure 1.

Assumptions. ProLoc integrates into existing mobile services that already collect location histories of devices. ProLoc requires devices to additionally collect and upload pseudonymous transmissions received over near-range radio from peer devices, and requires the service backend to implement the location proof algorithm (§3) and the defense against collusion attacks (§4). We note that these services that possess user location data can already infer periods of co-location; the addition of BLE data further corroborates these interactions without significantly expanding the platforms' existing knowledge on their users.

Devices are expected to upload their history within minutes of when events happened, effectively asking devices to commit to their trajectory and encounters immediately. Early commitment denies an adversary scope for certain retrospective attacks, where it rewrites the history of devices it controls to suit emerging objectives. The requirement is reasonable, because most smartphones are connected continuously, the bandwidth requirements are modest, and no user action is required. Late uploads of transmission received from peers are accepted, but an encounter can be used in a proof only if at least one of the peers uploaded the encounter in a timely fashion.

Our defense against collusion attacks relies on the detection of suspicious devices based on their connectivity to other devices. This fundamentally requires knowledge of some trusted devices as trust anchors. As we will show in §6, a very small number of devices is sufficient in ProLoc, and the set of devices can be different for each verifier. A verifier can choose their set freely from devices they trust, e.g., devices owned/operated by their own organization.

Power considerations. Location traces can be collected in a power-efficient manner using standard platform services. Sending and capturing BT advertisements in a power-efficient (and privacy-preserving) manner has been demonstrated in prior work [25, 34] and is now widely used in COVID-19 contact tracing apps.

2.1 Key Insights

The key insight behind ProLoc is that continuous collection, aggregation, and analysis of device locations and encounters can achieve properties not available in existing systems:

1) ProLoc can provide a feasible region for each instant along a device's trajectory *a posteriori*. To compute this region at instances when a device did not encounter other devices, ProLoc extrapolates from the closest previous and next confirmed locations and the maximal distance the device may have traveled in the interim. To compute this region, ProLoc intersects *isochrones* obtained from the OpenStreetMap [7] service. An isochrone bounds the set of locations reachable from a starting point within some period, in any direction, given the topology of the road and transportation network and the feasible speed of travel along each route.

2) ProLoc can mitigate retroactive collusion attacks involving fictitious devices. Because ProLoc requires devices to upload and thereby commit to their locations and encounters in a timely fashion, an attacker faces the difficulty of having to produce plausible device trajectories and encounters that afford its devices enough trust to appear legitimate, and are suitably positioned for an attack. Moreover, we sketch how the defense can be extended to cover *premeditated* collusion attacks as well.

3 PROLOC SERVICE

ProLoc's backend allows any participating device, called the *prover*, to prove a posteriori to a third-party, the *verifier*, that it was within a given region S at time t. The backend's only API call, $\texttt{prove\_loc}(d, S, t, TA, N)$, returns true when ProLoc is able to prove that device d was definitely within region S (specified as a bounding polygon on a map) at time t and false otherwise. The parameters S, t, TA and N are determined by the *verifier*. TA and N are needed for ProLoc's defense against collusion attacks (§4). TA is a (small) set of *trust anchors* –

devices that the verifier knows to be honest. As explained in §4, ProLoc uses TA to seed a TrustRank algorithm which identifies suspicious devices. N is the number of encounter peers which *independently* corroborate d's presence in S at time t. A higher N increases ProLoc's robustness to collusion attacks, but may also increase the sizes of regions S for which location proofs can be successfully found.

To answer the call $\texttt{prove\_loc}(d, S, t, TA, N)$, ProLoc's backend picks $M > N$ non-suspicious device peers which reported encounters with d close to time t. These peers, called *potential witnesses*, are denoted $w_1, ..., w_M$ here. For each potential witness w_i, the backend computes a region R_i in which d's presence at time t can be established using information from w_i alone: w_i's location history and its encounters with d. If R_i is contained in S, then the potential witness w_i is a *valid witness* for the proof. The call $\texttt{prove\_loc}(d, S, t, TA, N)$ returns true (successfully) if and only if the backend finds at least N valid witnesses, i.e., if the *union* of some N regions out of the M regions $R_1, \ldots, R_M$ is contained in S. The N valid witnesses form the proof's *quorum* and this union is called d's *feasible region* at time t.

The region S may be a circle specified as a center and radius, (θ, r). The API call $\texttt{prove\_loc}(d, (\theta, r), t, TA, N)$ then asks, "Can N devices' histories independently confirm that d was present within distance r of the point θ at time t?" In this case, we also call r the *precision*. Note that smaller r correspond to better precision.

For privacy reasons, a device may ask only for its own location proofs, i.e., a device may act as prover only for itself. However, ProLoc cryptographically signs the result (true or false) of the call together with the parameters to allow the device to prove its location to the verifier.

Next, we describe ProLoc's proof algorithm in detail.

Notation. We use the letter θ and its variants like θ', θ_w, etc. to denote points on a map. Similarly, we use t to denote time points, r to denote lengths, and R, S for regions.

Mapping service. ProLoc relies on a map service like OpenStreetMap [7]. This service provides a call $\texttt{isochrone}(R, T)$ which returns the entire region a device could have reached in time T starting somewhere in region R using any available means of transport at the fastest speeds possible.

ProLoc's algorithm. To answer the call $\texttt{prove\_loc}$ (d, S, t, TA, N), ProLoc first uses the *encounter selection algorithm* described later to select encounters $E_1, \ldots, E_M$ that d had with distinct peers in the vicinity of time t. Let $w_1, \ldots, w_M$ be the respective encounter peers (these are the potential witnesses). For each E_i, $1 \le i \le M$, ProLoc uses the function $\texttt{r\_peer}(E_i, d)$ described below to construct a region R_i in which d's presence at time t can be established from information provided by w_i alone. Next, for each i, ProLoc checks

Figure 2: Pseudocode of `prove_loc`. Dbase is ProLoc's internal database, and Map is the map service.

```
API call prove_loc(d, S, t, TA, N)
# Compute region for d w.r.t. a single peer location
function r_peer_loc(t_e, t_w, θ_w):
    R_{d_w}^{t_e} ← Map.isochrone({θ_w}, |t_e − t_w|)
    R_d^{t_e} ← Map.grow_region(R_{d_w}^{t_e}, r_BLE)
    return(Map.isochrone(R_d^{t_e}, |t − t_e|))
# Compute region for d w.r.t. a single peer encounter
function r_encounter(E_i, d):
    w ← E.other_peer(d)
    t_e ← E.timeof()
    (θ_w, t_w) ← Dbase.prev_location_report(d_w, t_e)
    (θ'_w, t'_w) ← Dbase.next_location_report(d_w, t_e)
    R ← r_peer_loc(t_e, t_w, θ_w)
    R' ← r_peer_loc(t_e, t'_w, θ'_w)
    R_i ← R ∩ R'
    return(R_i, w)
# Top level code
[E_1, ..., E_M] ← Dbase.encs_select(d, t, TA)
Q ← ∅  # Quorum;   R_F ← ∅  # Feasible region
for i in 1...M:
    (R_i, w_i) ← r_encounter(E_i, d)
    if R_i ⊆ S then { Q ← Q ∪ {w_i};   R_F ← R_F ∪ R_i }
if |W| ≥ N then return(true) else return(false)
```

whether R_i is contained in S or not. If it is contained, then ProLoc has found a valid witness. The algorithm stops with success if N valid witnesses are found, else it ends in failure. Figure 2 summarizes the algorithm.

Function `r_peer`(E_i, d). Let t_e be the time at which encounter E_i happened and let w be the encounter peer (the potential witness). Let θ_w and θ'_w be the self-reported locations of w preceding and following t_e, and let these locations be reported at times t_w and t'_w, respectively.

We start from one of d_w's location reports, say, θ_w at time t_w. We determine a possible region $R_{d_w}^{t_e}$ for d_w (not d) at the time of the encounter, t_e. Since w was at θ_w at time t_w, this region is simply $R_{d_w}^{t_e} = \text{isochrone}(\{\theta_A\}, |t_e - t_w|)$.

Next, we compute a possible region $R_d^{t_e}$ for d (not d_w) at time t_e. Since d encountered d_w at this time, d must have been within BLE range, say r_{BLE} meters, of d_w at time t_e. So, $R_d^{t_e}$ is obtained by expanding the region $R_{d_w}^{t_e}$ on the map by r_{BLE}. Then, we compute a possible region R for d at time t as $R = \text{isochrone}(R_d^{t_e}, |t - t_e|)$.

We repeat this process with d_w's second reported location θ'_w at time t'_w. This yields a second possible region R' for d at time t. Since d's location at time t is constrained by both R and R', the required region R_i is the *intersection* $R_i = R \cap R'$. Figure 3 illustrates this computation diagramatically.

Figure 3: Feasible region (shaded) w.r.t. one witness w.
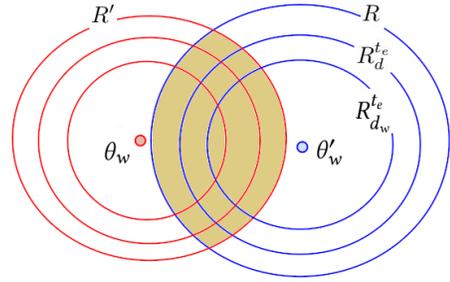
Encounter selection algorithm. The encounter selection algorithm, denoted `encs_select`(d, t, TA), selects encounters of d with distinct non-suspicious devices in the vicinity of time t and location θ. ProLoc first runs the TrustRank algorithm described in §4 seeded with the trust anchors TA to mark suspicious-looking devices. (ProLoc caches results of TrustRank. All subsequent calls with the same TA and, in particular, all subsequent calls from the same verifier reuse this result without recomputing TrustRank.)

Next, we find all encounters of d in a large time window $[t - a, t + a]$ centered at t (a is fixed to e.g., 5 mins). We remove encounters with suspicious-looking peers and sort the remaining encounters *ascending* by the following function f. The sorted list of encounters is the output of `encs_select`(d, t, TA).

$$f = |t_e - t| + \min(|t_e - t_w|, |t_e - t'_w|) + \frac{\min(|\theta - \theta_w|, |\theta - \theta'_w|)}{v}$$

Here, θ is the self-reported location of the device d at time t, t_e is the time of the encounter, θ_w and θ'_w are the reported locations of the encounter peer preceding and following the encounter, t_w and t'_w are the respective timestamps of these location reports, and v is the average travel speed in the vicinity of θ according to the map service. The function f embodies the fact that an encounter constrains d more if: (i) the time between t and the encounter is less (term $|t_e - t|$), (ii) the encounter peer has reported a location closer in time to the encounter (term $\min(|t_e - t_w|, |t_e - t'_w|)$), and (iii) the reported peer location is close to d's actual location θ (term $\min(|\theta - \theta_w|, |\theta - \theta'_w|)$).

Precision. The smallest region S that yields a successful proof depends on four factors: (1) N: Higher values of N provide more collusion resistance but prevent a successful proof for small S, (2) Local encounter density: If the proving device has many encounters in the spatio-temporal vicinity of the proof location, then the precision is likely to be better, (3) r_{BLE}: In places where BLE encounters are limited to small distances due to obstacles and noise, precision will be worse, and (4) Location reporting frequency: Precision is better when the proving device's peers report locations frequently.

In practice, r_{BLE} is determined empirically by measuring the effective BLE range in different conditions and choosing

an upper bound. Devices should be programmed to report locations at a fixed interval, also determined empirically.

Peer privacy. The success of a location proof reveals one bit of information about the witnesses' previous and next locations to the prover and the verifier. To prevent the leakage of witness trajectories using repeated proof calls, ProLoc rate-limits calls to its API from every prover.

4 DEFENDING AGAINST RETROACTIVE COLLUSION ATTACKS

An adversary's goal in attacking a LPS is to prove that a device it controls was at a target space-time location where it was actually not present, e.g., to fraudulently prove that they eye-witnessed an event. An attack is called *premeditated* if the adversary knows ahead of time where it will need a location proof later. To mount a premeditated attack, the adversary only needs a few real-looking colluding devices that – correctly or falsely – report presence at the target location and encounters with the target device around that location. Defending such attacks fundamentally requires additional infrastructure, as has been proposed in prior work (see §7). However, premeditated attacks are less relevant for use cases like citizen journalism, where the focus is often on unexpected events whose location and time the adversary cannot predict in advance.

Consequently, we focus on defending against the complementary class of *retroactive* attacks where the adversary cannot predict the space-time location of interest in advance. Mounting such an attack successfully requires continuous presence of adversarial devices at all locations of potential interest. In turn, this requires the adversary to control or simulate a very large number of devices.

Adversarial devices. To mount a retroactive attack, the adversary may deploy three kinds of (adversarial) devices. (i) *Corrupt* devices are real devices that the adversary controls at least partially. For a subset, it may even control physical whereabouts. (ii) *Sybil* devices, or Sybils, are virtual devices hosted on corrupt devices (e.g., rotating IDs on a phone). (iii) *Fictitious* devices, have no correspondence to an actual device. They are make pretend devices that report like real devices. The term *virtual* device includes both Sybil and fictitious devices. (In contrast, we use the term *honest* device for any device that is not controlled by the adversary.)

All types of adversarial devices can report false location trajectories and false encounters with other adversarial devices to enable (retroactive) attacks. Additionally, corrupt and Sybil devices can form real encounters with honest devices, but only along their real physical trajectories (the honest peer will not confirm a false encounter). Fictitious devices cannot form any encounters with honest devices.

Assumptions. Corrupt devices cost the adversary effort and money, so we assume that their number is limited. Virtual devices cost nothing, so we do not make any assumption on their number. Consequently, a successful defense against retroactive attacks must identify most virtual adversarial devices. The defense we present here does exactly this.

Key insight. ProLoc's defense relies on the insight that, even though the adversary can fabricate any number of virtual devices with arbitrary trajectories and encounters among themselves, any encounters between this virtual world and the world of honest devices are limited to corrupt and Sybil devices. Consider a graph of device encounters (Figure 1). In such a graph, the subgraphs representing the virtual devices and honest devices form clusters that are connected only by encounters involving corrupt and Sybil devices. Since we assume that the adversary is limited to a small number of corrupt devices and our defense limits the utility of Sybils (as explained later), these encounters will form a small cut between the adversarial and honest devices in the graph.

To arrive at a defense, we adapt the literature on random-walk-based approaches to detect virtual devices [9, 23] and ban them from quorums. We use a particular random walk, namely, the TrustRank algorithm [19]. We modify TrustRank to prevent Sybils from artificially increasing the cut between the real and virtual worlds. This allows us to recognize most fictitious devices immediately (i.e., our defense has high recall). Furthermore, our approach limits the number of Sybils an attacker can maintain without detection. In combination with a large enough N, this allows ProLoc to mitigate retroactive attacks effectively. While we did not design our defense for attacks where the adversary's goal is to reduce the precision of location proofs of *honest* devices, our defense does, in fact, prevent such attacks.

4.1 ProLoc's Defense

Recap of TrustRank. The original TrustRank algorithm [19] takes as input a directed graph with positive edge weights, and an initial set of trusted nodes (which we denoted TA in §3). It computes the answer to the following question: If (a) a random walk on the graph begins at a random trusted node, (b) at each step choosing to continue with a probability α, by traversing an outgoing edge with probability proportional to the edge's weight, then, what is the probability that it will end at any given node? This ending probability assigned to each node is the node's TrustRank score. Naturally, nodes connected to trusted nodes only via low-weight cuts have lower scores compared to nodes connected to trusted nodes via high-weight cuts. So, TrustRank distinguishes these two kinds of nodes from each other.

Our use of TrustRank. We apply TrustRank to the *encounter graph* whose nodes are devices and which has a directed edge from device g to d if g received at least one BLE advert from d. (Note that edges are oriented opposite to the direction of adverts.) We seed TrustRank with the set of trusted devices TA provided by the verifier. We first explain how we use this setup for our defense assuming there are no Sybil devices (the only virtual devices are fictitious), and then explain how we handle Sybil devices.

In the absence of Sybils, the width of the cut between the honest and fictitious devices is low since it is limited by the number of corrupt devices, which is assumed to be low. Hence, in the absence of Sybils, we can run the TrustRank algorithm described above with the edge weight from g to d set to the number of d's adverts received by g. This results in TrustRank scores with a bimodal distribution, with fictitious devices receiving significantly lower scores than real devices.

To actually classify devices as real or fictitious, we automatically determine a threshold that separates the two modes, and classify all devices with scores above the threshold as non-suspicious (likely real) and those below as suspicious (likely virtual). The threshold is determined by maximizing the sum of the percentages of true positives (real devices classified as non-suspicious) and true negatives (fictitious devices classified as suspicious) on a synthetic encounter graph that contains data from a set of devices which have been verified offline as being real (or not), and a set of synthetically generated attacker devices for different synthetic attack scenarios described in §6.2. We show in §6.2 that honest and fictitious devices are so well-separated by TrustRank scores on a real dataset that simulating a few attacks is actually sufficient for determining this threshold, even when the attack(s) occurring in reality are unknown.

Handling Sybils. Sybils can defeat the defense described so far: A small number of corrupt devices can broadcast the identifiers of any number of Sybil devices in repeated succession, thus increasing the cut-width between the real and fictitious devices to any extent. To defeat such attacks, we use a novel edge-weighting scheme that strongly penalizes large numbers of Sybils. We rely on the insight that Sybils on a single host appear to form concurrent encounters with nearby devices in such attacks. Our edge-weighting scheme heavily penalizes the contribution of concurrent encounters to edge-weights.

For each device, we divide the time series of received BLE adverts into epochs of fixed duration (e.g., 8 min). We call two adverts received by a device *concurrent* if they are in the same epoch. Next, we compute the weight of the edge from device g to device d, written $e(g, d)$, as follows: For each epoch in which g received an advert from d, we add to $e(g, d)$ 1 divided by the number of devices from which g received

concurrent adverts raised to a power L, for a fixed $L > 1$.[2]

$$e(g, d) \triangleq \sum_{E \in \text{epochs}} \frac{\text{if } (g \text{ recd. advert from } d \text{ in } E) \text{ then } 1 \text{ else } 0}{(\# \text{ devices from which } g \text{ recd. adverts in } E)^L}$$

To understand the effectiveness of the method, suppose that a corrupt device with m Sybils encounters a device g in an epoch where g receives adverts from t other devices. Then, each Sybil ends up getting an incoming edge weight $1/(t + m)^L$ from g in this epoch and the m Sybils together get an incoming edge weight of $m/(t + m)^L$ from g in this epoch. When $L > 1$, this function tends to 0 very quickly for large m. Hence, by creating a large number of Sybils m, the adversary lowers the incoming edge weights its Sybils receive from g in the epoch, relative to the incoming edge weights received from g by other devices in epochs where g encountered honest devices only. Since the total TrustRank score that the Sybils – and, hence, the entire virtual world – gets is proportional to the Sybils' total incoming edge weight relative to the total incoming edge weight of all devices encountered by g over time, it is in the adversary's interest to pick small m, i.e., to limit the scale of its Sybil attack.

We show experimentally in §6.2 that by setting $L = 3$, we can detect nearly all fictitious devices even in the presence of strong adversaries that corrupt up to 0.5% of real devices.

End-to-end defense. This leaves the adversary with only corrupt and Sybil devices at its disposal. Corrupt devices are limited in number by assumption. As for Sybils, the ideal strategy for an attacker in the face of our edge-weighting scheme is to rotate them at the granularity of epochs, in a way that they all get roughly equal TrustRank scores. If the average TrustRank score of real devices (including corrupt devices) is s and the TrustRank threshold determined above is T, then the adversary can sustain up to s/T Sybils per corrupt device with this strategy. In §6.2, we show that this ratio is around 10.6 for a real dataset. With this, if the adversary can corrupt c devices, it ends up with cs/T adversarial devices. In fact, we prove that the total TrustRank across all adversarial devices is bounded by a constant factor of the sum of the TrustRanks corrupt devices can collect:

THEOREM 4.1. *The sum of the TrustRank of attacker devices is at most proportional to the sum of the TrustRank of corrupt devices.*

PROOF. TrustRank of a device can be characterized as the probability that a random walker halts on the device, given:

- The walker begins on a (uniformly random) trusted device.

---

[2]This edge-weighting scheme is a slight simplification of our actual scheme, which uses continuous time and an integral instead of discrete epochs and a sum over epochs. Continuous time gets rid of epoch-boundary effects. The actual scheme is explained in appendix A .

- At each step, the walker proceeds with some probability α, and halts with probability $1 - \alpha$.
- At each step, if the walker proceeds, it steps to a neighboring device with probability proportional to edge weight (defined above).

Since trusted devices are real, and the only attacker devices that encounter real devices are corrupt devices, a random walker that halts on an attacker device must have (at some point) stepped to a corrupt device.
Therefore the probability that a random walker halts on an attacker device is, at most, the probability that a random walker steps to any corrupt device.

TrustRank is bounded by a constant factor of step probability. A random walker that steps to a device has at least a $1 - \alpha$ probability of halting on a that device. Therefore the TrustRank of a device is at least $1 - \alpha$ (which is a constant) times the probability that a random walker steps to that device.

Therefore, the probability that a random walker steps to any corrupt device is at most proportional to the sum of the TrustRank of corrupt devices. This in turn implies that the sum of the TrustRank of attacker devices is at most proportional to the sum of the TrustRank of corrupt devices. □

To limit how successfully an adversary can launch a retroactive attack, ProLoc additionally relies on the parameter N. With cs/T colluding devices as described above, the adversary can hold a quorum of N adversarial devices at no more than $cs/(T \cdot N)$ locations simultaneously. Just by way of example, an adversary that corrupts $c = 5$ devices, setting $N = 10$, with $s/T = 10.6$ as for our dataset, the adversary can cover only four locations at a time. To be successful in a retroactive attack, the adversary would have to correctly narrow its guess of the locations that could be relevant for an attack in the future to just a set of four.

Possible extension to premeditated attacks. Recall that in a premeditated attack, the adversary may move a quorum of adversarial devices to the target location on demand. Here, the small number of adversarial devices permitted by ProLoc's defense are sufficient for such an attack. While beyond the scope of this paper, the key to a defense against premeditated attacks that does not require infrastructure is to prevent the adversary from moving Sybils from the actual locations of the corrupt devices (which it cannot easily choose) to the target location, while maintaining their trust levels. One way to accomplish this is to diminish trust whenever a device moves in space-time, another is to tie trust to certain spatial zones of validity. A full design and evaluation of such a defense remains future work.

5 DATASETS

ProLoc takes as input the device history of each participating device. The device history is a time series of locations and BLE receptions from nearby devices (§2). In this section, we introduce the datasets we use in the experimental evaluation of §6: a simulated dataset and a real-world dataset.

Simulated dataset. We use a mobility simulator from prior work [11] together with OpenStreetMap data [7] to synthesize device locations and encounters in Copenhagen over a 15-day period. This involves three main steps: (i) we use the simulator to create a list of (hypothetical) users living in various Copenhagen neighborhoods, in accordance with Meta's publicly available data on population density, household, and age demographics [8]; (ii) we use OpenStreetMap data to identify places of interest (POIs) on the map, labeled according to their location type (e.g., supermarkets, schools, train stations); (iii) we use the simulator and Meta's publicly available mobility models (which provide the frequency of visits to each location type for different age groups) to synthesize individuals' mobility traces as a series of visits to POIs. A visit is a four-tuple consisting of a user device ID, a location or POI, a visit start time, and a visit end time. A user selects among POIs of the same location type with frequency inversely proportional to the distance between the user's home and the respective POIs—the so-called gravity model [21].

Location dataset. While at a POI, a user device records its location with a given average frequency, which we vary in our experiments from 1 to 5 minutes. Each record is a triple (device ID, location, timestamp). Devices do not record locations at home and in transit; this limited recording of locations is conservative as it makes our evaluation results worse than they would be had user devices continuously recorded locations. All the location records of a device constitute its location history.

Encounters. We cannot simulate individual BLE adverts because successful receipt of a BLE transmission is highly dependent on the physical environment, which is not covered in the mobility models we use. Instead, we adopt a conservative approach: we assume that if two devices remain within $r_{BLE} = 50$ m of each other for at least 5 minutes, then each of them will receive at least one BLE advert from the other, thereby establishing a mutual encounter. Our encounter dataset is the list of all such encounters represented as triples (device ID_1, device ID_2, timestamp), where timestamp is the midpoint of the time period during which the devices were in proximity.

In practice, not everyone will adopt ProLoc, so we generate mobility (location and BLE receipt) traces for only a chosen percentage of randomly chosen users (the *adoption rate*).

As an example, for an adoption rate of 20% and an average location reporting frequency of 3 minutes, our dataset has 128,882 unique participating devices (users), which together produce 72,905,340 location reports and 19,343,782 pairwise encounters. Over 15 days, each device participates in 150 encounters on average (min: 0, median: 106, max: 3,000) and provides 565.67 location reports on average (min: 33, median: 543, max: 2,670).

Real-world dataset. As mentioned above, we cannot simulate individual BLE adverts. Thus, to evaluate TrustRank, which relies on individual adverts, we use the SensibleDTU [5] dataset of BLE adverts from Danmarks Tekniske Universitet (DTU), Copenhagen.[3] The data was collected from March 2013 to August 2016 by DTU students carrying LG Nexus 4 phones, which supported Bluetooth 4.0 with A2DP. These devices collected BLE adverts using the OpenSensing [6] app for Android 4.2.2. In particular, the phones ran the data collector in the funf-v3 repo. The phones broadcasted unique device ids every 200ms, and processed all received adverts in batches. Phones periodically requested a new location from their location service and recorded them.

A total of 850 phones reported a total of 40.8M adverts received from other participating devices. Each entry includes the device ids of receiver and sender, and a timestamp. Each device recorded on average 48k adverts (min: 7, median: 38.4k max: 247.5k). Active devices wake up every 5 minutes and process any received BLE adverts, timestamping them at the moment of processing. Consequently, the distribution of time intervals between recorded batches for each device is very tightly clustered around 5min, with the 75th and 95th percentile at 5min and 20min, respectively.

6 EVALUATION

In this section, we present results of our experimental evaluation. We ran ProLoc on an Intel(R) Xeon(R) CPU E7-8857 v2 @ 3.00GHz machine (4 sockets, 12 cores/socket, 1 thread/core) with 1.48TB RAM, running Debian 11. TrustRank and the location proof algorithm are implemented in Python. Location proofs use OpenStreetMap graphs via the OSMnx 1.1.2 package [1]. Next, we show how location proof precision is affected by the encounter density and N. In §6.2, we show the effectiveness of ProLoc's defense against retroactive collusion attacks.

6.1 Location Proof Precision

Our objective is to understand how the location proof precision varies with the number of required verifiers N (higher N should reduce precision), the average location reporting

frequency of devices (higher frequency should improve precision) and the temporal encounter density at the location of the proof (higher encounter density should improve precision). A fourth factor, the ProLoc adoption rate, affects precision indirectly by shifting the distribution of temporal encounter densities: a higher adoption rate shifts the distribution towards higher encounter densities, and, hence, should result in improved precision on average.

We report on experiments with our simulated dataset, which is representative of individuals' mobility and encounters in an actual city. In all experiments, we ask for proofs that a device was within a circular region with a given center (which coincides with the location the device was visiting per our simulation) and a given radius R, which represents the precision. For each proof, encounter selection is set to find all encounters in a window of 10 minutes ($a = 5$ in the encounter selection algorithm). Our graphs show the minimum radius R (the highest precision) for which we could find a proof.

Effect of N. Figure 4 shows the 10th-, 50-th, and 90th-percentiles of the attainable precision radius R as a function of N in several different experimental settings (varying the average location reporting frequency, the encounter density and the adoption rate). The percentiles are estimated by running the algorithm of §3 on 5,000 randomly sampled visits in the respective experimental settings.

In all experimental settings, R increases with N, showing (as expected) that the precision reduces as the required number of independent verifiers increases.

Effect of the average location reporting frequency. Next, we explore the effect of the average location reporting frequency on the relation between R and N. Each graph of Figure 3a shows the 10th-, 50th- and 90th percentiles of R as a function of N for a different average location reporting frequency – 1 min, 3 min or 5 min. The adoption rate is fixed at 20% across the three graphs, and samples are drawn from visits with encounter densities close to the median density for this adoption rate (11 encounters in 10 minutes).

The distribution of precision radii (for any given N) shifts to the higher side as the frequency of location reports increases from 1 to 5 minutes. This is also expected because, when peer location reports bracketing an encounter are far apart in time, the location proof algorithm of §3 results in larger isochrones.

As an example, for an average location reporting frequency of 3 min, which is a reasonable trade-off between conserving device battery and proof precision, the median of the attainable precision varies from $R = 75$ m for $N = 1$ to $R = 500$ m for $N = 5$. This shows that proofs of reasonable precision are obtainable even with a 20% adoption and location reports as far apart as 3 min on average.

Effect of encounter density. Next, we consider the effect of encounter density. For this experiment, we fix the adoption rate and average location reporting frequency at 20% and 3 min, respectively. The three graphs of Figure 3b show the percentiles of R for samples drawn from visits with different encounter densities: 20, 40 and 50 encounters in 10 mins. As expected, for any given N precision radii are smaller (better precision) when the encounter density is higher.

Effect of adoption rate. Finally, we explore the effect of varying adoption rate on the relation between R and N. The three graphs of Figure 3c show the relation between R and N for three different ProLoc adoption rates – 10%, 20% and 40%. The average location reporting frequency is fixed at 3 min, and the encounter density is set to the median encounter density for the respective adoption rate. We observe that for a fixed N, R reduces as the adoption rate increases. This is in line with what we expect – as the adoption rate increases from 10% to 40%, the median encounter densities increase from 9 encounters in 10 minutes to 14 encounters in 10 minutes and the distribution of R shifts downwards (higher precision) accordingly.

Summary. From these experiments, we conclude that constructing location proofs with a precision of hundreds or even tens of meters is possible for reasonable values of N (upto 10), reasonable location reporting frequencies (3 minutes on average), and moderate adoption rates (20%).

6.2 TrustRank

Next, we evaluate the effectiveness of TrustRank in identifying fictitious and Sybil devices. Here, we use only the real-world dataset because our TrustRank algorithm relies on individual adverts, which are not available in our simulated dataset. Our real-world dataset includes only honest devices so, for this evaluation, we simulated several (in some ways optimally) powerful attacks on top of the real-world dataset.

Simulated attacks on the real-world dataset. In each attack, we first mark a number $c = 1, 2, 4$ or 8 of randomly chosen devices from among the 850 devices as *corrupt*. Since they are randomly chosen from honest devices, corrupt devices' behavior is fundamentally indistinguishable from honest behavior.

We then simulate m Sybil devices on each corrupt device, varying m from 1 to 128. We replicate every advert in the dataset whose source is a corrupt device to every Sybil on that device, thus simulating the Sybil attack of § 4, which gives the Sybils as many real encounters as possible.

Finally, for every device in our original dataset, we add an adversary-created *fictitious* device, which acts as its doppelgänger. The doppelgängers mimic the pairwise encounters

and reported locations of the real devices. This is a very powerful attacker that controls as many devices and uploads as much data as the real world. What's more, the behaviour of the fictitious devices is indistinguishable from that of honest devices. Without some trusted devices, there is no way to know which encounter subgraph is honest, and which is fictitious: they're symmetric.

Whenever a corrupt device receives an advert from a real device, it also receives an advert from the corresponding doppelgänger. Trust flows from the real world to the fictitious world along these bridging adverts through corrupt devices.

Overall, we simulate strong attacks where the attacker corrupts up to 8 real devices (1% of all devices), uses Sybils and uploads a fictitious world as large as the real world.

TrustRank. We randomly mark 10 devices as trusted, run TrustRank on the encounter graphs for each of the above scenarios (we empirically picked the exponent $L = 3$ for our edge weighting), and report our observations. (TrustRank is actually very robust to the initial choice of trusted devices so verifiers have a wide choice in the number and set of trusted devices they pick for location proofs. We experimented with different sets of trusted devices, ranging in size from 1 to 16, and the results were very similar in all cases.)

TrustRank threshold. We first examined the distributions of TrustRank scores of honest and attacker (corrupt, Sybil and fictitious) devices in different attack scenarios. As an example, Figure 4 shows the CDFs of the TrustRank scores of honest and attacker devices with $c = 8$, $m = 1$. There is a strong separation of over an order of magnitude between the TrustRank scores of honest and attacker devices.

Next, we determine a *uniform* cut-off threshold across all our attack scenarios (the vertical red line in Figure 4), by maximizing the sum of the true positive and the true negative rates across all attack scenarios (see § 4.1). For our dataset, this optimal threshold was 0.0000995. As shown later, this single threshold was very effective across *all* our attack scenarios. This indicates that, in practice, where the actual attack is unknown, one could still determine a useful threshold by simulating a limited number of possible attacks.

Above, we used ground truth about *all* honest devices to calculate the global threshold. In practice, the threshold can be approximated by adding simulated attackers to a subset of the encounter data limited to a *few* that are known to be honest. To confirm that this suffices, we recomputed the threshold on sample sub-datasets featuring only a small, randomly chosen subset of 25 honest devices (<3% of devices) and all the attack devices. This threshold was nearly as effective as our optimal threshold in separating honest and adversary devices.

Figure 4: Precision radius (R) as a function of the number of independent witnesses (N) for different location reporting frequencies, encounter densities and adoption rates. [Legend: 10-th percentile, median, 90-th percentile].

(a) Varying average location reporting frequencies: 1, 3 and 5 mins. Adoption rate fixed at 20%. Points in each graph sampled from visits with encounter densities close to the median for this adoption rate, which is 11 encounters in 10 minutes.
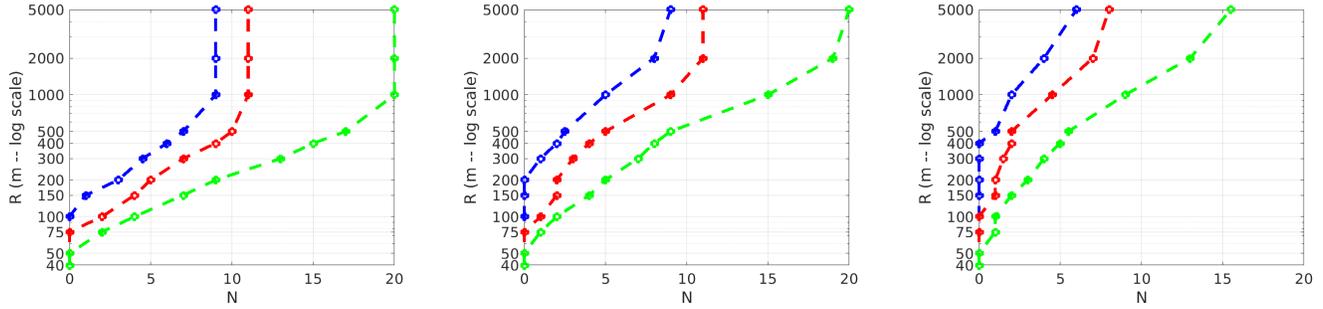
(b) Varying encounter densities: 20, 40 and 50 encounters in 10 minutes. Sampled from a simulation with adoption rate 20% and average location reporting frequency 3 minutes.
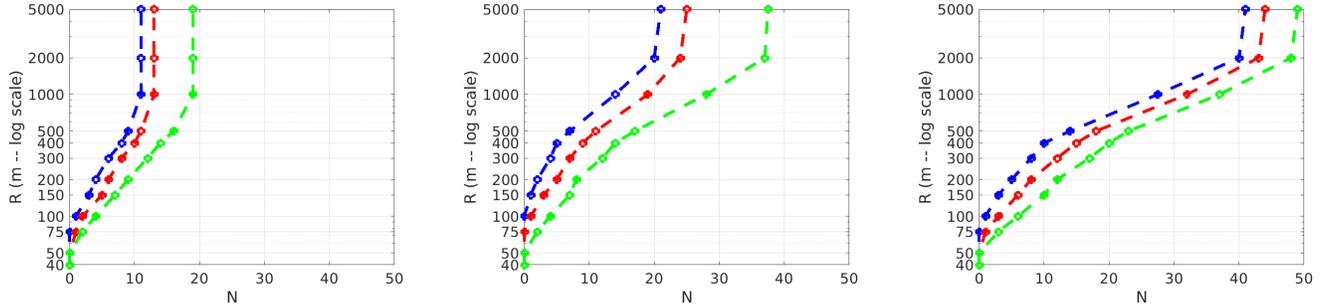
(c) Varying adoption rates: 10%, 20% and 40%. Average location reporting frequency fixed at 3 minutes. Points in each graph sampled from visits with encounter densities close to the median for this location reporting frequency and the respective adoption rate (9, 11 and 14 encounters in 10 minutes).
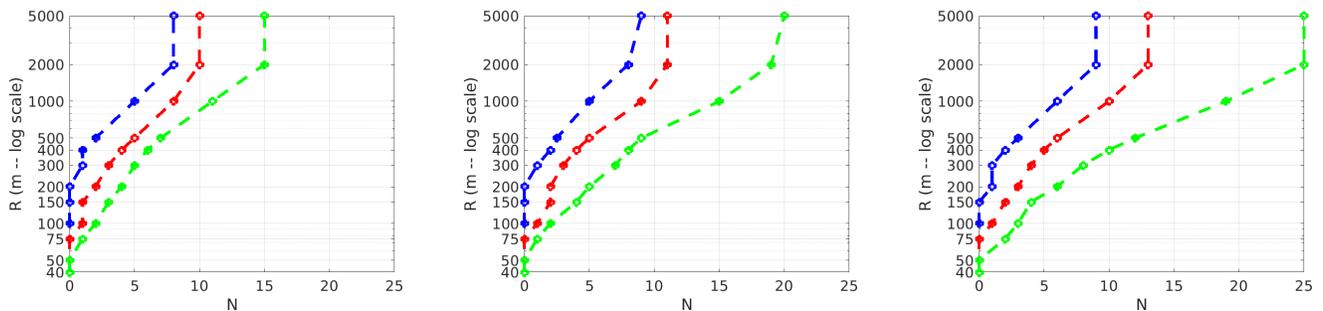
TrustRank effectiveness. Table 1 shows that with the threshold computed above, we can correctly identify most honest devices (column TP) and most virtual devices (columns TN), across all our attack configurations (we do not show results for $m > 16$ as those are nearly identical to those for $m = 16$). The fraction of correctly identified honest devices is consistently around 0.93, independent of the attack, again suggesting that our selection of a global threshold is robust.

The fraction of correctly identified fictitious devices is consistently above 0.998, even when $c = 4$ devices or nearly 0.5% of all real devices are corrupt. Even with $c = 8$ (nearly 1% corrupt devices), 0.957 of all fictitious devices can be detected. Thus, for adversaries that are limited to a small

Figure 4: CDF of TrustRank values for honest and attacker devices (corrupt and fictitious). Our threshold (the vertical red line) is computed using data from many simulated attacks.
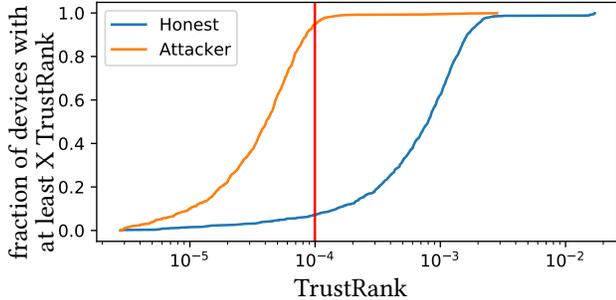
Table 1: TrustRank effectiveness with c corrupt devices, each with m Sybils. TP = fraction of correctly identified honest devices. TN = fraction of correctly identified virtual devices.

c	m	TP	TN	
			Sybil	Fictitious
1	1	0.929	0	1
	8	0.930	1	1
	16	0.930	1	1
2	1	0.930	0.5	1
	8	0.931	1	1
	16	0.931	1	1
4	1	0.929	0.25	0.998
	8	0.931	1	1
	16	0.931	1	1
8	1	0.927	0.125	0.957
	8	0.930	1	1
	16	0.930	1	1

fraction of corrupt devices, which we assume to be the case in our defense, TrustRank is nearly perfect at eliminating the threat of fictitious devices.

Finally, Table 1 shows that our edge-weighting scheme with $L = 3$ correctly discourages high numbers of Sybils per host by harming the adversary: With 8 or more Sybils per host, the TrustRank scores of all Sybils fall considerably, causing all virtual devices to be detected. This shows that our edge-weighting scheme works as intended.

As mentioned in §4.1, the ideal strategy of an adversary is actually to activate the Sybils on a corrupt device one at a time at epoch granularity, distributing trust as evenly as possible among them. Assuming that the average TrustRank of a real device is s and the TrustRank threshold is T, the number of Sybils an adversary can sustain per corrupt device without detection is s/T. In our experiments, this ratio is close to 10.6 in all attack scenarios, which indicates that it is not very sensitive to specific attacks.

6.3 Computational resources

We ran TrustRank over the DTU dataset plus the simulated attacks in under 1min on a single, standard server-class machine. As to the location proofs, their cost is dominated by the isochrone computation (about $N \times 23s$). However, the current computation is not optimized: it computes each isochrone from scratch relying on off-the-shelf Python APIs, and it does not store results that can be (partially) re-used to compute future location proofs in the same space-time region.

We believe that the computation can be considerably optimized for a production system, and scaled out easily. In fact, algorithms like TrustRank or PageRank are well-studied, and can run on enormous graphs (e.g., in production, TrustRank weights would be added to each edge incrementally, as each new data point arrives). Furthermore, the feasible region computation can be truncated once the feasible region grows beyond a pre-determined size (very large regions are not useful in practice), and the feasible region computation can also be trivially parallelized.

7 RELATED WORK

ProLoc differs from prior work on location proofs in two ways. First, all prior work provides proofs of presence at a point using interaction with fixed infrastructure or other devices *exactly* at the point of interest, which may not always exist. ProLoc instead provides proofs of presence in a (quantitatively sized) region using information about interaction with other devices close to the region of interest, not at a specific point, which is a more realistic requirement. Second, ProLoc provides a defense against (retroactive) collusion attacks on location proofs even when the adversary simulates virtual devices, a scenario that prior work did not consider.

Infrastructure-based location proofs. Saroiu et al. [31] articulate the importance of location proofs as enablers for reliable location-based services, and describe six applications. They sketch a method to produce location proofs using WiFi access points (AP). VeriPlace [26] also uses WiFi APs, but improves privacy by hiding clients' identities from APs and allows users to disclose approximate locations to services. Hasan et al. [20] augment WiFi AP-based location proofs with endorsements from nearby devices in BT range, to raise the bar for collusion attacks among devices and APs. A similar technique is described by Ferreira et al. [14].

CREPUSCOLO [13] relies on BT witnesses and trusted devices (TDs) placed at strategic locations that users are expected to encounter frequently. If a location proof has a TD as witness, the resulting proof is collusion-resistant. However, proofs for locations out of range of a TD rely on ordinary witnesses, and offer no protection from collusion.

Many prior works have addressed infrastructure-based location proofs in the context of vehicular networks [12, 18,

24, 37]. Pham et al. [30] and Maia et al. [27] describe location proofs for specific applications, namely proof of activity (e.g., distance/elevation traveled) and smart tourism (e.g., verifying that a tourist has visited certain sites), respectively.

In contrast, ProLoc does not require infrastructure for location proofs; it relies solely on peer devices' histories.

Infrastructure-less location proofs. Other prior work [10, 15, 28, 33, 35, 38] relies on BT witnesses and suggest or include collusion defenses. APPLAUS [38] suggests weighted witness quorums where a device's trust level depends on its history of witness choice, relative to the reported density of location proofs nearby in space and time. LINK's [33] per-device trust scores increase additively when a device votes with the majority in a successful proof, and diminish multiplicatively when the device acts suspiciously. STAMP [35] computes a per-device *entropy* value, which reflects the diversity and randomness in the device's witness selection and is additionally boosted whenever the device participates in a location proof that is confirmed by a trusted device. Arunkumar et al. [10] check if the near-range radio contacts and trajectories reported by all devices are consistent, and compute a trust score based on the ratio of encounters in which the device reports consistent information. A device's location is considered confirmed if the witness with the highest trust score agrees. PROPS [15] and PASPORT [28] do not consider collusion between witnesses and provers; their defenses work when either the witnesses or the prover is malicious. All these systems are vulnerable to attacks with fictitious devices, which can fabricate plausible trajectories, encounters, witness selections, and even proofs involving trusted devices using a small number of corrupt devices and Sybils, thus acquiring trust to the point where they are indistinguishable from honest devices.

Hardware-based location proofs Some prior work relies on a hardware root-of-trust (e.g., TPM modules) on mobile devices to ensure the integrity of location readings [16, 29, 32]. Some work has addressed GPS spoofing in specific contexts like aviation (Crowd-GPS-Sec [22]) and mobile games [36]. Gonzálex-Tablas et al. [17] provide a survey of location authentication protocols and spatial-temporal attestation services. In contrast to these, ProLoc does not require trusted hardware on the devices.

8 CONCLUSION

ProLoc enables a mobile service to compute retroactive location proofs by integrating devices' trajectories and encounters. ProLoc computes a feasible region, including the locations at which the device could plausibly have resided at the time, and it does so in a principled manner from the set of encounters the device had around the time and its peers' trajectories. Moreover, ProLoc mitigates large-scale

collusion involving fictitious devices and Sybils, effectively limiting the adversary's power to a small multiple of the corrupt physical devices it controls. As a result, even a powerful attacker is very limited in its ability to retroactively generate false proofs. Finally, we sketch how to extend ProLoc's defenses to premeditated attacks, although a full treatment of such defenses remains as future work.

A EDGE-WEIGHTING SCHEME

In §4.1, we described an approximation of our actual edge-weighting scheme for defeating Sybil multiplicity attacks. Here, we describe the actual scheme. This scheme also uses a time window w (typically 8 minutes), but it does not divide time at each node into discrete epochs to distribute edge weights. Instead, it considers infinitesimally small epochs over intervals $[t, t + dt]$ at each node g. In each such epoch, g gives some weight to the outgoing edge towards every device d from which g has received an advertisement in the vicinity of t, specifically, in the interval $[t - w/2, t + w/2]$. The weight given to each such edge in the epoch is $dt/(\# \text{ such devices})^L$. The total weight of the outgoing edge towards d is the *integral* over time of these infinitesimally small weights.

One additional difference from what we described in §4.1 is that we cap the weights of edges to prevent any device from getting too much trust from a single device. Figure 5 summarizes our revised edge-weight function, $e(d, g)$.

REFERENCES
[1] Boeing, G. 2017. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. Computers, Environment and Urban Systems 65, 126-139. doi:10.1016/j.compenvurbsys.2017.05.004 . https://github.com/AMDESE/AMDSEV/tree/sev-snp-devel. Accessed: 2021-12-20.

[2] Britannica. Citizen journalism. https://www.britannica.com/topic/citizen-journalism. Accessed: 2023-11-15.

[3] ProLoc: Technical Report. https://tinyurl.com/proloctr. Accessed: 2022-03-25.

[4] YaleGlobal Online. Citizens Fight Back with Cellphones and Blogs. https://archive-yaleglobal.yale.edu/content/citizens-fight-back-cellphones-and-blogs. Accessed: 2023-11-15.

[5] Sensible DTU Project. https://github.com/SocialComplexityLab/sensible_docs, 2018. Accessed on 28/12/2020.

[6] OpenSensing GitHub. https://github.com/OpenSensing, 2020. Accessed on 13/01/2020.

[7] OpenStreetMap. https://www.openstreetmap.org, 2021. Accessed on 19/08/2021.

[8] Data for Good at Meta. https://data.humdata.org/organization/meta?, 2023. Accessed on 27/11/2023.

[9] Lorenzo Alvisi, Allen Clement, Alessandro Epasto, Silvio Lattanzi, and Alessandro Panconesi. SoK: The Evolution of Sybil Defense via Social Networks. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 382–396. IEEE Computer Society, 2013.

$$e\left(d,g\right) \triangleq \min\left(\text{weight\_cap}, \int_{\text{time } t} \frac{\left(\begin{cases} 1 & \text{if } g \text{ recd. advert from } d \text{ in } [t - w/2, t + w/2] \\ 0 & \text{o/w} \end{cases}\right) \mathrm{d}t}{\left(\begin{array}{c} \text{\# devices from which } g \\ \text{recd. adverts in } [t - w/2, t + w/2] \end{array}\right)^{L}}\right)$$

Figure 5: Edge-weighting scheme of §4.1 in continuous time

[10] S. Arunkumar, M. Srivatsa, M. Sensoy, and M. Rajarajan. Global attestation of location in mobile devices. In *MILCOM 2015 - 2015 IEEE Military Communications Conference*, pages 1612–1617, 2015.

[11] Gilles Barthe, Roberta De Viti, Peter Druschel, Deepak Garg, Manuel Gomez-Rodriguez, Pierfrancesco Ingo, Heiner Kremer, Matthew Lentz, Lars Lorch, Aastha Mehta, and Bernhard Schölkopf. Listening to bluetooth beacons for epidemic risk mitigation. *Scientific Reports*, 12, 2022.

[12] Mohamed Baza, Mahmoud Nabil, Mohamed Mohamed Elsalih Abdelsalam Mahmoud, Niclas Bewermeier, Kemal Fidan, Waleed Alasmary, and Mohamed Abdallah. Detecting Sybil Attacks using Proofs of Work and Location in VANETs. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2020.

[13] Eyüp S. Canlar, Mauro Conti, Bruno Crispo, and Roberto Di Pietro. CREPUSCOLO: A collusion resistant privacy preserving location verification system. In *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*, pages 1–9, 2013.

[14] João Ferreira and Miguel L. Pardal. Witness-Based Location Proofs for Mobile Devices. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–4, 2018.

[15] Sébastien Gambs, Marc-Olivier Killijian, Matthieu Roy, and Moussa Traoré. PROPS : A PRivacy-preserving lOcation Proof System. In *Proceedings of the IEEE 33rd International Symposium on Reliable Distributed Systems*, pages 293–307, 2014.

[16] Peter Gilbert, Landon P. Cox, Jaeyeon Jung, and David Wetherall. Toward Trustworthy Mobile Sensing. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, HotMobile '10, page 31–36, New York, NY, USA, 2010. Association for Computing Machinery.

[17] A. I. González-Tablas, K. Kursawe, B. Ramos, and A. Ribagorda. Survey on location authentication protocols and spatial-temporal attestation services. In Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuanshun Dai, and Laurence T. Yang, editors, *Embedded and Ubiquitous Computing – EUC 2005 Workshops*, pages 797–806, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[18] Michelle Graham and David Gray. Protecting Privacy and Securing the Gathering of Location Proofs – The Secure Location Verification Proof Gathering Protocol. In Andreas U. Schmidt and Shiguo Lian, editors, *Security and Privacy in Mobile Information and Communication Systems*, pages 160–171, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[19] Zoltan Gyongyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *Proceedings of the 30th international conference on very large data bases (VLDB)*, 2004.

[20] R. Hasan and R. Burns. Where Have You Been? Secure Location Provenance for Mobile Devices. *ArXiv*, abs/1107.1821, 2011.

[21] Kingsley E Haynes and A Stewart Fotheringham. Gravity and spatial interaction models. 2020.

[22] K. Jansen, M. Schäfer, D. Moser, V. Lenders, C. Pöpper, and J. Schmitt. Crowd-GPS-Sec: Leveraging Crowdsourcing to Detect and Localize GPS Spoofing Attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 1018–1031, 2018.

[23] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Random Walk Based Fake Account Detection in Online Social Networks. In *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017, Denver, CO, USA, June 26-29, 2017*, pages 273–284. IEEE Computer Society, 2017.

[24] Qinglei Kong, Feng Yin, Yue Xiao, Beibei Li, Xuejia Yang, and Shuguang Cui. Achieving Blockchain-based Privacy-Preserving Location Proofs under Federated Learning. In *ICC 2021 - IEEE International Conference on Communications*, pages 1–6, 2021.

[25] Matthew Lentz, Viktor Erdélyi, Paarijaat Aditya, Elaine Shi, Peter Druschel, and Bobby Bhattacharjee. SDDR: Light-Weight, Secure Mobile Encounters. In *Proceedings of the 23rd USENIX Security Symposium*, pages 925–940, 2014.

[26] Wanying Luo and Urs Hengartner. VeriPlace: A privacy-aware location proof architecture. pages 23–32, 01 2010.

[27] Gabriel A. Maia, Rui L. Claro, and Miguel L. Pardal. CROSS City: Wi-Fi Location Proofs for Smart Tourism. In Luigi Alfredo Grieco, Gennaro Boggia, Giuseppe Piro, Yaser Jararweh, and Claudia Campolo, editors, *Ad-Hoc, Mobile, and Wireless Networks*, pages 241–253, Cham, 2020. Springer International Publishing.

[28] Mohammad Reza Nosouhi, Keshav Sood, Shui Yu, Marthie Grobler, and Jingwen Zhang. PASPORT: A Secure and Private Location Proof Generation and Verification Framework. In *IEEE Transactions on Computational Social Systems*, pages 293–307, 2020.

[29] H. Othman, H. Hashim, M. A. Yuslan Razmi, and J. Ab Manan. Forming Virtualized Secure Framework for Location Based Services (LBS) using Direct Anonymous Attestation (DAA) protocol. In *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 622–629, 2010.

[30] Anh Pham, Kévin Huguenin, Igor Bilogrevic, and Jean-Pierre Hubaux. Secure and Private Proofs for Location-Based Activity Summaries in Urban Areas. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, page 751–762, New York, NY, USA, 2014. Association for Computing Machinery.

[31] Stefan Saroiu and Alec Wolman. Enabling new mobile applications with location proofs. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, pages 1–6, 2009.

[32] Stefan Saroiu and Alec Wolman. I Am a Sensor, and I Approve This Message. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, HotMobile '10, page 37–42, New York, NY, USA, 2010. Association for Computing Machinery.

[33] Manoop Talasila, Reza Curtmola, and Cristian Borcea. LINK: Location Verification through Immediate Neighbors Knowledge. In Patrick Sénac, Max Ott, and Aruna Seneviratne, editors, *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 210–223, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[34] Lillian Tsai, Roberta De Viti, Matthew Lentz, Stefan Saroiu, Bobby Bhattacharjee, and Peter Druschel. enClosure: Group Communication via Encounter Closures. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 353–365,

2019.

[35] X. Wang, A. Pande, J. Zhu, and P. Mohapatra. STAMP: Enabling privacy-preserving location proofs for mobile users. *IEEE/ACM Transactions on Networking*, 24(6):3276–3289, 2016.

[36] Shing Ki Wong and Siu Ming Yiu. Detection on GPS Spoofing in Location Based Mobile Games. In Ilsun You, editor, *Information Security Applications*, pages 215–226, Cham, 2020. Springer International

Publishing.

[37] Yifan Zhang, Chiu C. Tan, Fengyuan Xu, Hao Han, and Qun Li. VProof: Lightweight Privacy-Preserving Vehicle Location Proofs. *IEEE Transactions on Vehicular Technology*, 64(1):378–385, 2015.

[38] Z. Zhu and G. Cao. APPLAUS: A privacy-preserving location proof updating system for location-based services. In *2011 Proceedings IEEE INFOCOM*, pages 1889–1897, 2011.