CoCoT: Collaborative Contact Tracing

Trevor Kann
tkann@cmu.edu
Carnegie Mellon University

Lujo Bauer
lbauer@cmu.edu
Carnegie Mellon University

Robert K. Cunningham
robertkcunningham@pitt.edu
University of Pittsburgh

ABSTRACT

Contact tracing can limit the spread of infectious diseases by notifying people of potential exposure to disease. Manual contact tracing is resource-intensive, but much of it can be automated using mobile phones, which are ubiquitous and can detect and record nearby contacts. Two major problems arise with automated contact tracing (ACT): preventing abuse for mass surveillance and accurately determining contacts. For example, the most widely adopted solution—Google and Apple's Exposure Notification (GAEN)—protects user privacy but suffers from inaccurate distance measurements that result in poor risk assessments. We propose to use collaboration among nearby devices to increase distance estimation accuracy, and therefore risk assessment accuracy, while minimizing the loss of user privacy. Our protocol, CoCoT, extends GAEN, a proximity-based, distributed ACT protocol, by adding an additional broadcast to share locally-derived distance estimates. To evaluate CoCoT, we develop a method for merging phone sensor datasets with human interaction datasets to approximate realistic scenarios and test our protocol. CoCoT improves distance estimate accuracy by 28% over the current best distance estimators and we analytically show impact on privacy, security, and battery consumption are minimal.

CCS CONCEPTS

• **Theory of computation** → **Distributed algorithms**; • **Applied computing** → Health care information systems; • **Security and privacy** → **Privacy-preserving protocols**; **Distributed systems security**; • **Human-centered computing** → *Smartphones.*

KEYWORDS

Automatic Contact Tracing, Distributed Optimization, Network Localization, Smartphone Privacy

ACM Reference Format:
Trevor Kann, Lujo Bauer, and Robert K. Cunningham. 2024. CoCoT: Collaborative Contact Tracing. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy (CODASPY '24), June 19–21, 2024, Porto, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3626232.3653254

1 INTRODUCTION

Manual contact tracing (MCT) has proven useful for slowing disease spread. When an infected person is identified—the index case—their recent contacts are located, isolated, and tested. If any test positive, they become new index cases and the process repeats [46].

MCT comes with several challenges [7], including: the long time required for trained professionals to thoroughly investigate, the availability of said professionals, and the accuracy of index cases' recollections. These problems are exacerbated when cases increase and diseases are more infectious, making MCT difficult to scale.

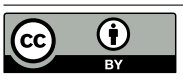
During the COVID-19 pandemic, many recognized that these shortcomings could be ameliorated by leveraging the prevalence of smartphones to implement Automatic Contact Tracing (ACT). ACT can identify proximate devices and notify users if a recent contact tests positive, after which they could self-isolate and test themselves. ACT does not aim to replace MCT, but to assist health workers by automating much of the contact tracing process [48].

Early ACT implementations could be abused for mass surveillance [27]. Several solutions were proposed that better preserve user privacy [4, 10, 23, 29, 38, 55]. Currently, Google and Apple's Exposure Notification (GAEN) [23, 29] is the most widely adopted ACT protocol in the USA and Europe [45]. GAEN is a decentralized, privacy-preserving protocol that uses proximity to record contacts.

One challenge with GAEN is that proximity is derived from Bluetooth Low Energy's (BLE) received signal strength indicator (RSSI), a notoriously bad distance estimator, impacting GAEN's accuracy [62]. RSSI is an unreliable indicator of distance because BLE signals are reflected off materials common in urban environments, such as glass and metal. Several approaches were proposed to improve distance-estimation accuracy, but all come with significant drawbacks to privacy, battery life, or contact-tracing utility. Broadly, approaches to improve distance estimates fit into three categories: using additional on-phone sensors (e.g., accelerometers [49]), augmenting physical infrastructure (e.g., wireless beacons to localize devices [63]), or collaboration between devices (e.g., sharing information with others [24]). We focus on collaboration and how it can be practically used to increase ACT's distance-estimation accuracy.

In this paper, we offer an addition to the GAEN protocol—*CoCoT*—to improve distance estimates and we evaluate its benefits. CoCoT adds a single additional communication between phones to collaboratively localize themselves among proximate phones, improving distance estimates. CoCoT trades off a small amount of privacy, security, and battery life for a significant increase in accuracy.

To test our collaborative algorithms, we need a dataset containing both locations of people and pairwise BLE measurements between them. However, at this time, no such dataset exists. To overcome this challenge, we created a dataset by combining two existing datasets containing either locations or BLE measurements but not both. We developed a simulation that combines locations from a cocktail-party location dataset [1] and simulates pairwise BLE measurements based on a BLE dataset [36]. Using this constructed dataset, we show the benefits of collaboration are significant: distance estimate errors can be reduced by up to 28% and false positives/negatives by 21%. We also demonstrate that the losses of privacy, security, and battery life are small.



CODASPY '24, June 19–21, 2024, Porto, Portugal

ACM ISBN 979-8-4007-0421-5/24/06
https://doi.org/10.1145/3626232.3653254

Some proposals to improve distance estimates [49] only indicate whether two phones are within 6 feet of each other; in contrast, CoCoT outputs precise distance estimates between devices, allowing for more flexibility with risk assessment than approaches that only estimate if phones are within a pre-specified cutoff distance [49]. This makes our protocol potentially more useful, since guidelines about safe distances sometimes change with better disease understanding or epidemiological landscape changes [20].

2 RELATED WORK

We first briefly discuss the history, design choices, and trade-offs of different ACT solutions (Section 2.1). We focus on proximity based ACT—specifically GAEN, a widely adopted standard—and one of its key technical challenges: accurately estimating inter-personal distances. We present work in network positioning and optimization that tackles similar problems (Section 2.2) and then discuss improvements that have and could be applied to ACT (Section 2.3).

2.1 Automated Contact Tracing

Multiple ACT designs have been developed and deployed, differing based on local requirements and preferences. Some of their main design choices and their trade-offs are summarized in Table 1. Most ACT implementations can be categorized as either location or proximity based, and either centralized or distributed. For comparisons beyond our summary, see PURE [14].

2.1.1 Location Based ACT. One approach uses a Central Authority to determine contacts based on absolute locations of phones collected with GPS [27]. Phones regularly collect and upload their location to a central authority. When a new index case is found, the central authority examines the index case's location history to identify other nearby users, then alerts them of potential exposure.

2.1.2 Proximity Based ACT. Some ACT designs improve privacy by using *proximity* instead of location. One of these, BlueTrace, uses BLE to identify other nearby phones [4]. The central authority gives phones unique IDs that are broadcast frequently (chirps) for other, nearby phones to record. When an index case is found, their phone uploads a list of recorded chirps to the central authority. The central authority identifies which phones broadcast these chirps and informs their users of potential exposure. With this design, an honest but curious central authority is unable to learn the locations of users. However, it still learns which users are near each other, possibly revealing private information about user's contacts.

To further increase user privacy, decentralized approaches such as *Decentralized Privacy-Preserving Proximity Tracing* (DP-3T) [55] were proposed. These also use BLE chirps to identify proximate phones, but locally determine contacts to prevent the central authority from knowing whom users were near. When a new index case is found, their phone uploads its previously broadcast chirps to the central authority. Other phones download a list of chirps belonging to index cases daily and determine locally if any were proximate, potentially exposing the user to illness. The most widely adopted of these decentralized solutions is GAEN [44, 45].

Other recently proposed solutions (e.g., [10]) focus on further increasing user privacy. Our paper is concerned with the distance estimates between devices and could be integrated with these more privacy-focused solutions. Due to its prevalence, we focus on GAEN.

Google Apple Exposure Notification (GAEN). GAEN's features are built in to recent Android and iOS OSs. To perform GAEN [23, 29], phones create a cryptographically random *Temporary Exposure Key* (*TEK*) at the start of each day. This *TEK* is used to derive a *rolling proximity identifier* (*RPI*) roughly every 10 minutes. Phones continuously chirp (BLE broadcast) these *RPI*s every 200-270 ms. Other nearby phones record these *RPI*s alongside an estimated separation distance and timestamp. Later, phones download a list of infected users' *TEK*s and determine if any sensed *RPI*s originated from them. If a match is found and considered a significant risk (based on distance), the user is notified to self-isolate and test themselves.

Determining contacts locally and anonymously uploading *TEK*s makes it impossible for an honest but curious central authority to learn anything more than the number of reported index cases. The protocol's cryptography makes it difficult for users to de-anonymize index cases or generate false positives by broadcasting random *RPI*s.

GAEN defines significant risk using several parameters chosen per implementation, including proximity (distance apart) [18, 23, 29]. Proximity is currently estimated via BLE RSSI, but BLE RSSI is a notoriously bad distance estimator [62]. This makes measuring proximity, and therefore risk, a key technical challenge for GAEN. Furthermore, researchers have identified that GAEN's perceived accuracy is a key factor for user adoption [37] and high adoption-rates are critical for any ACT solution's success [26]. However, the same study identified that any ACT solution must also balance battery consumption, and user privacy and security [37].

2.2 Network Positioning

Distributed networks historically solved similar problems: given noisy, unreliable distance estimates between devices, compute a most-likely network layout [24]. Also similar to ACT, (most) devices lack sensors to obtain their absolute location (e.g., GPS) and must rely on external information gathered through the network to position themselves within the network. While some assumptions conflict, these algorithms can act as inspiration to improve ACT.

Generally, network positioning algorithms are centralized or distributed. Centralized algorithms attempt to gather sufficient information for a single (powerful) device to position every other device [8, 61]. Acquiring this information can be difficult if the device lacks easy communication channels to every other device. Alternatively, distributed, iterative algorithms can achieve positioning consensus over time across most or all devices [24]. These *gossip-like* algorithms slowly come to consensus, requiring many communications, often on the order of the number of devices in the network [9]. We elaborate on algorithmic specifics in Section 3.3.

2.3 Improvements to ACT Distance Estimates

ACT uses BLE RSSI to detect contacts (Section 2.1.2) however BLE RSSI alone is insufficient for use as a distance estimator [39, 62]. Network positioning algorithms demonstrate the usefulness of using additional information to improve distance estimates, which ACT might be able to utilize. Unfortunately, GAEN cannot directly use centralized or decentralized network positioning algorithms (Section 2.2) due to conflicting requirements: Centralized algorithms

Design Aspect	Options	Tradeoffs	Examples
Positioning	Location: GPS, cell towers; Proximity: Bluetooth, audio	Location vs. privacy; also accuracy	Israeli Contact Tracing [27] vs. BlueTrace [4]
Architecture	Centralized vs. Distributed	Simplicity/utility vs. privacy (to central authority)	BlueTrace [4] vs. DP-3T [55]
Primary beneficiary	Society vs. Individual	Societal hotspots vs. individual cases	South Korea's health alerts [32] vs. GAEN [23, 29]
Installation	Mandatory vs. voluntary	Forced enrollment vs. individual control	Hong Kong's LeaveHomeSave[28] vs. NOVID [38]

Table 1: ACT design choices and implications. Different countries selected different approaches, appropriate for their societies.

require sending potentially private information to a central authority for accurate positioning and/or contact tracing, something GAEN explicitly avoids. Distributed solutions typically rely on fixed-position devices that can use several rounds of communication to locate an object [24]. Since ACT is built for people moving with their phones, these algorithms are not generally applicable.

Additional information for use in distance estimation can come from either local or remote sources, and either from humans or devices. We categorize information sources into three categories: local cases we categorize as *local sources* (Section 2.3.1), additional remote devices as *infrastructure* (Section 2.3.2), and remote humans as *collaboration* (Section 2.3.3).

2.3.1 Local Sensors. Modern phones are equipped with several sensors that can accurately sense their environment and be leveraged to better estimate distances between users. Sensor measurements can supplement BLE RSSI to improve local distance estimates and then be forgotten, helping preserve users' privacy.

One example of how additional sensors can be used has been showcased by NIST's *Too Close for Too Long* (TC4TL) challenge [49]. TC4TL tasked competitors to produce a model to best estimate interpersonal distances given BLE RSSI readings alongside other phone sensors, like accelerometers and gyroscopes. Competitors found the accuracy of BLE-RSSI-based distance estimates were significantly improved by additional sensors [2, 25, 51].

A limitation is that phone sensors can only measure their own environment. For example: two variables affecting BLE-based distance estimates are whether phones are indoors or outdoors [58], and the phone's pose (e.g., held in a hand vs. in a pocket) [3, 25]. If a phone determines it is outdoors, other nearby phones are likely also outdoors and with similar BLE profiles, where *nearby* is defined as within radio range. However, if it determines that it is held in a hand, it has little understanding of other, nearby phone poses.

As another alternative, phones could rely on users as additional information sources, asking them about details of recent contacts to assist in assessing risk. Examples include asking if someone is wearing a mask or behind a wall. Acquiring this information requires the user to be an active and accurate participant; misremembering critical information may lead to inaccuracies.

2.3.2 Additional Infrastructure. Additional external infrastructure could be developed to facilitate accurate contact tracing. For example, BLE beacons could act as an indoor GPS alternative [40, 63]. In crowded areas, several beacon devices could be positioned to accurately triangulate phones, providing them with relative distances to nearby devices, even those unsensed by BLE [22]. Beacons might also broadcast environmental statistics, like air quality measurements and whether the beacon is indoors or outdoors, to better asses risk between users [5, 59]. This approach could provide phones with additional information without using significant battery power.

One drawback is that deploying such infrastructure would be costly in both time and money. Also, allowing beacons to accurately position or detect users might incur privacy issues. For example, if a beacon with a known location detects a phone, a central authority with beacon access could learn that user's absolute location. Privacy preserving workarounds may exist, but beacons controlled by a central authority intrinsically pose a privacy and security risk, since users cannot verify the devices are honest and well-behaved.

2.3.3 Proximate Phone Collaboration. Another way for a phone to collect additional information is by collaborating with nearby phones. As previously mentioned, it is difficult for a phone to locally estimate another phone's pose (e.g., in a hand vs. a pocket); however, each phone can accurately estimate its own pose [3, 25] and share this information with nearby devices. Examples of other similarly shareable information include device characterization, indoor vs. outdoor estimates [58], or distance estimates (which we use). Existing ACT protocols can readily use collaboration, making deploying it more economical than expanding physical infrastructure.

Collaboration's drawbacks include potential privacy risks, disinformation attacks, and additional battery consumption. For privacy, naïvely broadcasting information might reveal private information. For security, malicious users may broadcast intentionally false information [34], making honest users' results less accurate, possibly worse than if they had never collaborated. Finally, any extra broadcasts from the device will inherently use power.

In this paper, we use collaboration to improve ACT, while minimizing risks to privacy, security, and additional power usage.

3 PROPOSAL: IMPROVE DISTANCE ESTIMATES BY COLLABORATION

In this section, we first describe our threat model, including how it relates to existing ACT and network-positioning threat models (Section 3.1), followed by our notation definitions (Section 3.2). Then, we describe the core contribution of our paper: several methods to improve distance-estimation accuracy in ACT by sharing initial distance estimates among neighbors (Section 3.3). We call these algorithms CoCoT, for collaborative contact tracing.

3.1 CoCoT's Threat Model

Like GAEN, we wish to accurately notify at-risk contacts of index cases while minimizing private information leakage. The trade-off between privacy and utility has been a primary focus of ACT literature [4, 10, 15, 23, 29, 38, 39], and we continue that work here.

Also like GAEN, adversaries are assumed to have full knowledge of CoCoT and control some nearby commercially available BLE-enabled devices (like smartphones) to eavesdrop on BLE packets or broadcast custom-made packets. However, adversaries are unable to alter the way their radio fundamentally behaves. Adversaries are also assumed to have access to the central authority's information. Adversaries' goals may include degrading user privacy or decreasing the algorithm's accuracy and utility. To date, attacks on GAEN exploited compromised BLE-enabled applications to eavesdrop on GAEN's BLE transmissions [17]. These vulnerabilities have since been fixed [41]: currently, the information sent and received by GAEN is accessible only by the privileged OS (apps still interface with the OS to perform GAEN); an adversary would need to have access to the OS's BLE software/drivers to mount this attack.

Information that we consider private is user identity, infection status, current and past location and contacts. GAEN makes this information difficult to determine, but collaboratively sharing information may make this information easier to obtain. We quantify how much information an adversary can learn in Section 6.

Decreasing the accuracy of nearby phones' risk assessments could make users over or under-cautious with higher rates of false-positives or false-negatives, respectively. Doing so may decrease the utility of CoCoT and decrease users' trust in the system, lowering adoption rates [37]. With GAEN, this requires adversaries modifying how their BLE transceiver behaves, but with collaboration this could be done by reporting dis-information [34]. We demonstrate how CoCoT performs against this attack in Section 7.

3.2 Notation

Assume a set of N phones $\mathcal{P} = \{p_1, p_2, ..., p_N\}$. For notational simplicity, p_i and i are used interchangeably. Algorithms are shown from the perspective of phone p_i but run identically on each phone. Each phone $p_i \in \mathcal{P}$ has a two-dimensional coordinate $\mathbf{x}_i \in \mathbb{R}^2$ (representing their position) and a sensing range r_i (within which they can detect and communicate with all phones), both unknown to the phone. Phones symmetrically in sensing range of each other are called *neighbors*, and we define the set of p_i's neighbors as:

$$\mathcal{N}(p_i) = \{p_j \in \mathcal{P} : \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \min\{r_i, r_j\}\} \tag{1}$$

Using the GAEN protocol, each phone p_i computes a noisy distance estimate to each of its neighbors p_j:

$$\delta_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|_2 + \varepsilon \tag{2}$$

where ε is measurement noise.

Honest phones share the goal of estimating distances $d_{i,j}$ to each of their neighbors $p_j \in \mathcal{N}(p_i)$ as accurately as possible, where $\delta_{i,j}$ is the initial measurement, $d_{i,j}$ is the final distance guess. Phones can broadcast any information they chose to accomplish this goal, but anything shared is broadcast to all neighbors within range.

Some algorithms also produce a relative position estimate vector $\mathcal{X}_{i,j} \in \mathbb{R}^2$ locally on p_i's device to each neighbor p_j. These vectors correspond to p_i's guess of p_j's position relative to itself: $\mathcal{X}_{i,j} \approx \mathbf{x}_i - \mathbf{x}_j$, as if p_i had a 2-dimensional mapping of its surroundings. From these vectors, p_i can calculate the distance between two estimates using the euclidean distance $d_{l,j} = \|\mathcal{X}_{i,l} - \mathcal{X}_{i,j}\|_2$. Due to privacy concerns, we do not broadcast the estimates $d_{i,j}$ or position vectors $\mathcal{X}_{i,j}$, unlike network positioning algorithms [9, 16, 24].

3.3 Candidate CoCoT Algorithms

CoCoT's addition to GAEN is a protocol for refining the initial, BLE-RSSI-derived distance estimates. In particular, after each phone p_i estimates its distance to any neighbor p_j, it broadcasts that estimate, $\delta_{i,j}$, to all its neighbors. Each phone subsequently uses the received distance estimates to calculate more accurate distance estimates to each of its neighbors.

We propose three algorithms that could be used to collaboratively improve distance estimates between phones, starting with the simplest algorithm and progressing to more complex and accurate algorithms. All algorithms only broadcast initial distance estimates to their neighbors, they do not broadcast any other information. First, we define the *Average* algorithm, which averages the two distances measurements between users. Then, we define two graph-theoretic algorithms—*Graph Drawing* and *Spring-Mass*—that take advantage of the topology defined by their neighbors to yet more accurately estimate distances.

I think I wo For analysis, we assume that phones are synchronized with instantaneous, reliable communication. In practice, collaborative distance estimates only need to be made soon enough after δ is recorded so that the relative positions of each phone have not changed much. We discuss these assumptions and other practical considerations in Section 8.

3.3.1 Average. The *Average* algorithm reduces measurement noise by averaging the two distance estimates between the phone p_i and its neighbors. p_i gathers its neighbors' initial distance estimates to itself and outputs an updated estimate $d_{i,j} \leftarrow \frac{1}{2}(\delta_{i,j} + \delta_{j,i})$.

3.3.2 Graph Drawing. We observe that public interactions usually include more than two people, all standing on relatively flat ground. Recall that phones broadcast to all their neighbors so, when p_i broadcasts its distance estimates, it shares an entire list of its neighbors and their associated distance estimates, defining a graph (or topology). *Graph Drawing* leverages the topology defined by its neighbors to improve individual distance estimates to each of its neighbors. The improvement is because, intuitively, a topology of neighbors provides more measurements, or constraints, than only a single edge (used by *Average*). These additional constraints can reduce noise and provide relational information.

Graph Drawing uses multi-dimensional scaling (MDS) [8] to estimate relative position vectors $\mathcal{X}_{i,j} \in \mathbb{R}^2$ to all neighbors that best preserve distance estimates. MDS is traditionally used to map higher-dimensional data to lower-dimensional spaces while preserving relative distances between data points but is also useful for finding layouts of points when only pairwise distances are known. MDS requires distance measurements between every point; however, in our case, some of a phone's neighbors will not detect each other, especially if they are at opposite ends of the phone's sensing area (*Average* did not require a dense graph; it only needed distance estimates from its neighbors to itself). To account for these missing

distances, *Graph Drawing* borrows the missing-distance-estimation scheme from the *As Rigid As Possible* (ARAP) algorithm [61], described below. We chose ARAP for its simplicity and because it does not require additional information or communication; it can be performed entirely locally. Once missing distances are estimated, MDS generates position vectors $\mathcal{X}_{i,j}$ for p_i's neighbors, which are used to compute improved relative distances $d_{i,j}$.

ARAP estimates distances between two nearby phones that do not sense each other but share a neighbor. First, ARAP bounds missing distances from below and above: suppose phone p_i is neighbors with p_j and p_k, but p_j is not a neighbor of p_k. p_i knows that if p_j and p_k were within each other's sensing range they would be neighbors, so they must be further apart than either r_j or r_k; their distance is at least: $\min\{r_j, r_k\} \leq \|\mathbf{x}_j - \mathbf{x}_k\|_2$. None of the phones know the value of either range r, so it too can be bounded by the furthest neighbor: $\max_{p_l \in \mathcal{N}(p_j)}\{\delta_{j,l}\} \leq r_j$; and similarly for r_k. p_i also knows that the furthest apart p_j and p_k could be is if they were co-linear with p_i, upper-bounding their distance. Generalizing, the triangle inequality tells us p_j and p_k can be no further apart than the shortest two-hop path between them: $\|\mathbf{x}_j - \mathbf{x}_k\|_2 \leq \min_{p_l \in \mathcal{N}(p_j) \cap \mathcal{N}(p_k)} \left(\delta_{i,l} + \delta_{l,k}\right)$. In summary, the missing distance is bounded by:

$$min\{r_j, r_k\} \leq \|\mathbf{x}_j - \mathbf{x}_k\|_2 \leq \min_{p_l} \left(\delta_{i,l} + \delta_{l,k}\right) \tag{3}$$

The missing distance estimate is approximated as the distance halfway between the bounds:

$$\delta_{j,k} \leftarrow \frac{1}{2}\left(min\{r_j, r_k\} + \min_{p_l} \left(\delta_{i,l} + \delta_{l,k}\right)\right) \tag{4}$$

This estimate is repeated for any neighbors of p_i that are not themselves neighbors, making $\mathcal{N}(p_j)$ a complete graph. Then, as mentioned above, MDS is used to estimate position vectors $\mathcal{X}_{i,j}$ to each neighbor. Neither the estimated missing distances nor the position vectors are shared to other phones.

3.3.3 Spring-Mass. Finally, we propose a custom-weighted spring-mass stress minimization algorithm: *Spring-Mass*. Graph-based algorithms like *Graph Drawing* position phones relative to each other in a dense graph with nearby measurements affecting one another, pushing and pulling other measurements into configurations where the measurements agree more. If the initial measurements have relatively little noise, *Graph Drawing* improves their accuracy using the additional information provided by the graph. However, *Graph Drawing* treats every measurement with equal weighting, even though not all measurements are equally accurate (see Section 4.2). Hence, a few highly inaccurate measurements can make *Graph Drawing*'s accuracy much worse than if *Graph Drawing* was not used at all; the *Spring-Mass* algorithm fixes this by assigning different weights to each measurement to rely more heavily on accurate measurements than inaccurate ones.

To estimate the accuracy of a measurement (and hence to assign the appropriate weight to that measurement), we consider two factors: the *separation* of (i.e., distance between) the two phones and the *discrepancy* between their measurements of this distance. When two phones are farther apart, their distance estimates are statistically less likely to be accurate (see Section 4.2). Additionally, further apart phones have less topological information in common, since their sensing areas overlap less, and they should impact each other's estimates about other phones less. Hence, as the separation of phones increases, the influence of their initial distance estimate on the computed topology should decrease.

For the discrepancy, if two measurements of the same quantity are not equal, then at least one must be incorrect. The greater the discrepancy between the two measurements, the more likely it is that the measurements are not an accurate representation of the distance between two phones. Hence, as discrepancy between measurements increases, the measurements' impact on the topology decreases too. *Separation* is used in previous work on network positioning [24], but *discrepancy* is a novel contribution of ours.

Spring-Mass runs after *Graph Drawing* and utilizes *Graph Drawing*'s missing distance and position estimates as initial inputs. To make some measurements more impactful than others, *Spring-Mass* uses a classic physics optimization problem: the spring mass system; specifically, a network positioning variant [24]. Phone p_i simulates a spring between every pair of phones in $\mathcal{N}(p_i)$. The spring's resting length is the measured distance between the two phones. Each spring's stiffness is defined by a spring constant k that estimates measurement accuracy. *Spring-Mass* then updates position estimates to minimize the total potential energy of the springs.

Stiffer springs (larger k) impact the positioning of phones in a neighborhood more than weaker springs. Ideally, measurements with more accuracy will be relied on more or, equivalently, have the largest spring constants k. We calculate k using a positive function f of the separation, discrepancy, and a bias term that adjusts how much the separation and discrepancy terms impact k. Specifically, the input to f is a weighted sum of a constant (bias) and the separation and discrepancy terms of the measurements. The weightings are tunable hyperparameters: w_b, w_δ, and w_d for the bias, separation, and discrepancy, respectively. We calculate discrepancy as either $\Delta_{l,j} = |\delta_{l,j} - \delta_{j,l}|$ or $\left|\frac{\delta_{l,j} - \delta_{j,l}}{\delta_{l,j} + \delta_{j,l}}\right|$, the choice of which is also a hyperparameter. For the bias to produce a non-constant change in k, f needs to be non-linear, such as $f(x) = e^{-x}$. Finally, k is calculated between every phone p_l and p_j:

$$k_{l,j} = f\left(w_b + w_\delta \frac{\delta_{l,j} + \delta_{j,l}}{2} + w_d \Delta_{l,j}\right) \tag{5}$$

To select the best hyperparameter values, we performed a grid search; details can be found in our technical report [31]. With k computed, we iteratively update the positional estimates $\mathcal{X}$ to find a layout that minimizes the *stress* of the system, which is defined as the total sum of potential energies of the springs:

$$stress = \sum_{p_j, p_l \in \mathcal{N}(p_i)} \sum k_{l,j} \left| \frac{\delta_{l,j} + \delta_{j,l}}{2} - d_{l,j} \right|^2 \tag{6}$$

where $d_{l,j} = \|\mathcal{X}_{i,l} - \mathcal{X}_{i,j}\|_2$, as shown before in Section 3.2. We minimize stress in the same way as Gotsman and Koren [24].

4 EVALUATION METHODOLOGY

To test our algorithms, we need a dataset containing both human positions (or topology) and BLE-RSSI-derived distance estimates between them. Presently, no such dataset exists. Therefore, to create the necessary dataset, we contribute a method to combine human-interaction datasets and BLE RSSI datasets. First, we discuss the human-interaction datasets (Section 4.1) and the phone sensor

dataset (Section 4.2) we used. Then, we discuss how we combined the two datasets to create a new, joint dataset to test our proposed algorithms using a simulation (Section 4.3). Finally, we describe how we quantitatively compare each algorithm (Section 4.4).

4.1 Positioning Datasets

To realistically sample inter-personal distances, we require a realistic distribution of human locations during regular interactions. We use both a real, measured human-interaction dataset as well as a hand-made, synthetic dataset to test our algorithms' performances across a variety of circumstances. From these human interaction datasets, we can extract positions of people over time.

For our real, measured dataset we use the cocktail-party half of the *SALSA* dataset [1], which records participants moving and interacting over 30 minutes. Among human-interaction datasets, SALSA has a uniquely large participant count ($N = 18$) with significant movement between social groups throughout the study. Alternative datasets lack high participant count, are too static, too dispersed, or contain scripted interactions [35, 53, 57, 60].

To test our algorithms' generalizability and because other human-interaction data is scarce, we hand-made several additional test scenarios. These scenarios are modeled on a cafeteria, a conference room, and a banquet hall. The cafeteria has two rows of three rectangular tables of 6 feet × 2.5 feet, seating six people each; the conference room has four of the same rectangular tables in one row with one extra person at each end; and the banquet hall has two rows of round tables, 5 feet in diameter, sitting eight people each. We also have a sparse version of each setting: for the cafeteria and banquet-hall,we remove all but the two most distant tables; for the conference-room, we remove every other person. We describe these scenarios more precisely in our technical report [31].

4.2 BLE RSSI Dataset & Distance-Estimation Model

Given the location and true distances between people, we need to sample realistic distance estimates between them. We use the dataset provided in NIST's TC4TL competition [49] for approximating the distributions of distance estimates using a machine learning model. We use this dataset because of its use of additional on-phone sensors, giving us a state of the art RSSI-to-distance models.

NIST used the MIT Lincoln Laboratory procedure [15] to collect BLE RSSIs along with other phone sensors' data to create a dataset for the TC4TL challenge [49]. The dataset contains data from 28 smartphone models recorded from different positions on user's bodies (in hand, pant pocket, etc.) and in both indoor and outdoor environments [48]. At 3, 4, 5, 6, 8, 9, 10, 12, and 15 feet apart, phones measured and recorded BLE RSSIs between them and other local sensor values[1]. The data comes in a log-file format containing BLE/sensor samples at different sample rates, appearing sequentially with time stamps; each log file is labeled with a true separation distance. From the provided datasets, we use the MITRE Range-Angle dataset [36], as it has the most realistic interactions[2].

[1]Sensor data includes accelerometer, gyroscope, attitude sensor, gravity sensor, altimeter, and magnetic field sensors; the estimated current heading and activity; and BLE RSSI. Data also includes true distance apart, transmitting and receiving devices' model and user pose, and transmitting power.

[2]Other datasets were artificially noiseless, such as in an anechoic chamber

4.2.1 Training the Distance Estimation Model. The TC4TL competition showed that using additional sensors could make BLE-RSSI-derived distance estimates more accurate. We train our baseline neural network model on the TC4TL dataset, building on the experiences of of TC4TL competitors [2, 25, 51]. We preprocess TC4TL data by converting each BLE interaction sample to a sequence of vectors. To get a uniform sampling rate across sensors, we upsample slower sensors to the fastest sampling rate by copying their most recent reading whenever the fastest sensor records a new sample. For more detail, see our technical report [31].

In total, we obtained 3600 transformed samples. These samples were split into a validation set (1000 samples) and a training set (2600 samples) that we used to train a recurrent neural network to estimate separation distances. We tested several machine-learning architectures and hyperparameters to find a combination that best predicted the separation distances in the validation set. We initially used mean squared error for the loss function, but this model produced outliers, which decreased the accuracy of the collaborative algorithms. Mean fourth error proved to be a better loss metric: $loss = |estimate - reference|^4$ because it more strongly discouraged outliers. When searching different model parameters, we stopped training when cross-validation loss began to plateau. A shallow GRU-based model [13] proved best. The model had 220 GRU cells that fed into a dense linear layer, with a single output estimating the separation in feet. Finally, we evaluated the validation set on this model and recorded the outputs binned by reference distance.

TC4TL competitors found that discrete models (only predicting within 6 feet or not) achieved the best scores. However, our collaborative algorithms required continuous distances to function. Nevertheless, we can compare our model's score to theirs. Our continuous model's leaderboard test score was 0.49, while the TC4TL competitors scores ranged from 0.41 to 0.60 (lower is better) [42]. Because our scores were similar to other competitors, we believe our model approximates a state-of-the-art distance estimator.

4.2.2 Extending Distance Estimate Dataset. The size of the BLE dataset is significantly smaller than the human interaction datasets[3]. To avoid repeatedly using the same, few BLE datapoints when combining datasets, we extrapolated the finite BLE datapoints into a continuous distribution that can be sampled infinitely. We approximate the BLE dataset's distance-estimate distribution by performing a density estimate (Gaussian kernel, $\sigma = 0.5$ft) on the BLE datapoints binned by true distance apart. Density estimates are integrated and normalized to create a cumulative density function (CDF) of distance-estimate distributions that we can efficiently sample from when combining datasets.

4.3 Combining Datasets

To combine the datasets, we construct a discrete-time, agent-based simulation. Simulated people are positioned based on the human-dataset, then BLE-RSSI-derived distance estimates are sampled based on proximity. At the start of the simulation, each agent's broadcast/detection range r is set to 15 feet, matching the BLE dataset [36]. At each time step, agents are positioned in a two-dimensional field based on the human-interaction data (Section 4.1).

[3]The BLE dataset is only 3600 datapoints, the SALSA dataset has roughly 500 timesteps, with 18 participants requiring $18 \times 17 = 306$ samples per timestep

Then, each agent i determines its neighbors j within range and locally estimates distances to each, drawn from the distance estimate CDFs (Section 4.2.2), and records it as $\delta_{i,j}$. Once all initial, local distance estimates have been made, agents share these estimates with their neighbors. Agents evaluate each algorithm to generate an updated, collaborative distance estimate d to each neighbor per algorithm. These ds are stored to evaluate our algorithms (discussed below in Section 4.4). Once agents run every algorithm, the simulation proceeds to the next time step. Due to our simulation's randomness, every experiment is run 15 times using the same set of seeds. We made our tool and sample datasets publicly available.[4]

4.4 Metrics

To determine the best of the candidate algorithms (Section 3.3), we measure the mean absolute error and standard deviation of estimate errors, as well as NIST's *nDCF* score [49]; lower is better in every metric. Mean error measures how far from the truth our algorithm's estimates are, but can be misleading if under-estimates are equally incorrect as over-estimates; measuring absolute error accounts for this. The *nDCF* score gives a different sense of how often our algorithms are incorrect by measuring how often false positives/negatives are recorded. We record metrics for both the real and synthetic human-interaction datasets, but optimize for the real dataset and test for generalization on the synthetic datasets.

The *nDCF* score is defined as

$$nDCF_{cutoff} = \frac{\mathbf{P}[false\ positive] + \mathbf{P}[false\ negative]}{2} \tag{7}$$

Where *cutoff* is evaluated at 3, 6 (default), or 10 feet to obtain $nDCF_3$, $nDCF_6$, and $nDCF_{10}$ scores, respectively. *nDCF* scores are slightly misleading, however, as our human interaction dataset has many interactions close to 6 feet. This means small errors in updated distance estimates can lead to higher (worse) $nDCF_6$ scores despite being more accurate measurements on average. For this reason, we optimize for mean absolute error instead of *nDCF* scores.

5 EVALUATION RESULTS

Here, we compare the baseline, non-collaborative local estimates to our collaborative algorithms and show that collaborative algorithms perform better (Section 5.1). Then, we demonstrate the best performing algorithm, *Spring-Mass*, generalizes well (Section 5.2).

5.1 *Spring-Mass* Algorithm Performs Best

All three algorithms and the baseline local estimates were tested in a benign environment; the results are summarized in Table 2, with the best results in bold. The *nDCF* scores displayed are on the simulated dataset, not the TC4TL dataset; they cannot be directly compared with the *nDCF* scores from the TC4TL competition and serve only as a metric to compare between our tested algorithms.

All collaborative algorithms outperformed local estimates except for *Average*'s $nDCF_3$ score. Algorithms other than *Spring-Mass* did not significantly affect *nDCF* scores. *Spring-Mass*, notably, reduced mean absolute error by 28%, standard deviation by 24%, as well as reduced *nDCF* scores, reducing false positives/negatives by 21%. A one-tailed paired T-test shows the improvement is statistically significant: we reject the null-hypothesis that the mean absolute error of the collaborative estimates is equal to or greater than local estimates ($t = -20.1$, $p < 0.001$).

The overall trend in our results is expected, as each consecutive algorithm increases in ability to reduce measurement errors. Somewhat unexpected is that the *Graph Drawing* algorithm does not reduce errors more than the *Average* algorithm. However, this is because measurements with significant error have a strong effect on all neighbors, impacting the accuracy of the entire neighborhood instead of just between the two phones, as explained in Section 3.3.3. Reducing the impact of these distance estimates by using weaker springs allows the *Spring-Mass* algorithm to perform best.

5.2 *Spring-Mass* Generalizes Well

To see if our best algorithm, *Spring-Mass*, generalizes well we also evaluate it on several synthetic scenarios. Note, we optimized the hyperparameters of *Spring-Mass* exclusively on the recorded SALSA dataset. Table 3 shows the results of running *Spring-Mass* on these new topologies. For interactions with low average distance between users, *Spring-Mass* performs similarly well as on our real dataset, with up to 30% improvement in mean absolute error and 31% improvement in standard deviation. Improvement tapers off as the interaction distances grow larger. This trend is expected since *Spring-Mass* benefits most from dense, nearby connections and will eventually perform the same as GAEN as interactions grow sparser.

6 PRIVACY ANALYSIS

CoCoT uses the same infrastructure as GAEN and only affects the way phones broadcast, so attacks that exploit GAEN's infrastructure or initial broadcasts are unaffected by CoCoT but discussed nonetheless (Section 6.1). Sharing information does have novel privacy implications, however, and can benefit nearby adversaries. We discuss how sharing information might help adversaries reconstruct a topology (Section 6.2) and briefly discuss how CoCoT also increases sensing area (Section 6.3).

6.1 Privacy Risks Inherited from GAEN

While GAEN was designed to be privacy preserving for users, some attacks still exist [17]. To learn private information (Section 3.1) about nearby users, an attacker must first receive a victim's *RPI*(s) and match that *RPI*(s) with the victim's identity [17]. For example,

Algorithm	MAE	SD	$nDCF_3$	$nDCF_6$	$nDCF_{10}$
local estimate	3.07	4.03	0.88	0.67	0.74
Average Out	2.61	3.30	0.93	0.63	0.73
Graph Drawing	2.75	3.52	0.84	0.66	0.72
Spring-Mass	**2.21**	**3.05**	**0.75**	**0.53**	**0.65**

Table 2: Spring-Mass performed best of the tested algorithms by all metrics. Mean absolute error (MAE) and standard deviation (SD) measure how close to truth our algorithms' predictions are; *nDCF*, measures false positive/negative rates for determining if phones are within a cutoff distance of 3, 6, or 10 feet, respectively. Lower is better in all metrics.

[4] https://github.com/pwwl/CoCoTsimulator

a malicious phone p_a may capture some of honest p_h's broadcast *RPI*s and associate them with p_h's owner. Later, the attacker could check the central authority to see if any of the uploaded TEKs generate the captured *RPI*s, informing the attacker that p_h's owner is infected. *RPI* matching is critical to learn any of the private attributes mentioned in Section 3.1 for both GAEN and CoCoT.

To match users with their *RPI*s, attackers must have some knowledge of nearby users' location and identity. For this analysis, we assume the attacker can (visually) see the nearby users, and so identify them and ascertain their relative positions (i.e., the physical topology). Other methods of sensing users' locations and identities could be used instead, and are equivalent for this analysis.

With GAEN, this information permits matching nearby users with their *RPI* if and only if they are a unique distance to the attacker(s). If there are multiple users that are equidistant to the attacker, then the attacker cannot distinguish between them. In Figure 1, for example, a GAEN attacker could not distinguish between the two blue phones (p_1 & p_2) because they are equidistant.

6.2 Privacy Risks of Topology Reconstruction

Sometimes, CoCoT makes matching users to their *RPI*s easier. The previously mentioned physical topology, which is based on what attackers can visually observe, includes information about user identities (to the extent these are known to the attacker) and users' physical positions. With CoCoT, and particularly when using *Spring-Mass*, attackers learn pairwise distances between neighboring phones. These can be used to construct another topology, which we call the *CoCoT topology*. If attackers can overlay the CoCoT topology and the physical topology, they can match nearby users with their *RPI*s.

Configuration	People Per Square Foot	% Improvement MAE	% Improvement SD
SALSA	0.06	28.01	24.32
Cafeteria	0.15	28.17	30.05
Conference	0.16	30.56	31.66
Banquet Hall	0.11	11.50	17.31
Sparse Cafeteria	0.06	10.44	15.45
Sparse Conference	0.08	13.95	14.79
Sparse Banquet Hall	0.04	13.70	18.80

Table 3: CoCoT performs best on datasets where people are close together. SALSA has a particularly high score because our hyper-parameters are optimized for it. Shown is percent improvement of mean absolute error (MAE) and of standard deviation (SD) by *Spring-Mass*.

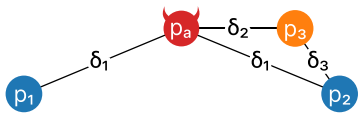


Figure 1: In some cases, CoCoT's information sharing provides weaker privacy compared to GAEN. Using GAEN, an attacker p_a cannot distinguish between phones p_1 and p_2 because they are equidistant to p_a. Using CoCoT, p_a can distinguish them because of p_2's unique proximity to p_3.

Overlaying topologies will not always succeed: CoCoT only knows pairwise distances between neighbors, which could produce a topology consistent with a rotated or mirrored physical topology. The CoCoT topology is always constrained translationally relative to the physical topology because the attacker knows where they themselves are in the topology. But, the CoCoT topology is not constrained rotationally nor reflectively by default. If the topology is asymmetric, the attacker can find phones in the topology to further constrain it rotationally and reflectively. Figure 1 shows an example: p_3 is a unique distance away from the attacker, p_a, so the topology is rotationally constrained. The attacker can physically observe that p_2 is a close neighbor of p_3, constraining the topology reflectively. Now, the attacker can match all nearby users' *RPI*s to their identities because the topology is asymmetric. If the topology is symmetric, CoCoT would offer no advantage over GAEN to an attacker. We next delineate under what circumstances CoCoT's topology benefits an adversary; we summarize in Table 4. We also provide additional examples in our technical report [31].

Non-Stationary Topology. If the topology changes over time, then an adversary can almost always[5] match all nearby users with their *RPI* from moving relative to them. Monitoring the topology as it moves over time effectively makes every user's distance to the adversary unique. For example, if two phones are equidistant to the attacker, like in Figure 1, moving closer to either of the two phones would cause one's signal strength to increase more than the other, allowing the adversary to distinguish phones. Recording distances over time is sufficient to make the topology asymmetric, making most boxes in Table 4's mobile attacker columns gray. In general, whenever users are all not-equidistant, all users can be matched to their *RPI* regardless of CoCoT. CoCoT provides no benefit over GAEN to an attacker that can move.

The remainder of this analysis therefore assumes an entirely stationary topology. A stationary topology might occur if an attacker only captures a single moment in time. While this is unrealistic, we explore it to understand privacy implications.

Three or More Receivers. If the adversary has three or more receivers, they can triangulate users using traditional techniques, since every point in the plane is a unique distance to the three receivers. Similarly to why a single malicious phone produces a translationally constrained topology, three malicious phones produce a topology that is further constrained rotationally and reflectively, since the attacker can distinguish where their own phones are in the topology. Thus, in Table 4's "3+ receivers" columns, rows which are symmetric or equidistant/mixed are gray. CoCoT provides no benefit over GAEN to attackers with three or more receivers.

Two Receivers. Usually, two receivers can also triangulate users in the same way three receivers can, with one exception. Two receivers cannot triangulate users in the same way three receivers can if other users are positioned reflectively symmetric about the reflective axis. In this case, the entire topology is symmetric and CoCoT offers attackers no benefit over GAEN. If any sensed users are not positioned reflectively symmetrically then an attacker could

[5] See our companion technical report [31] for edge-case analyses.

use CoCoT's topology to match every user to their *RPI*. If some, but not all, users are pairwise symmetric, then CoCoT gives an attacker an advantage over GAEN.

Single Receiver. CoCoT provides the most benefit over GAEN to an attacker with a single, stationary receiver. As before, the topology needs to be asymmetric for attackers to use CoCoT's topology to match nearby users with their identity. In this case, CoCoT gives attackers an advantage over GAEN when at least some users are equidistant to the attacker's receiver; GAEN cannot distinguish between them but CoCoT can. An example is seen in Figure 1.

Summary. CoCoT can provide an advantage to a stationary attacker with fewer than three receivers. We argue that an attacker could get these advantages without CoCoT by using more powerful receivers, more receivers, or moving relative to their environment/recording over time. Additionally, in reality, it is unlikely multiple users would be stationary, equidistant from an adversary.

Topology Properties				***Attacker Properties***					
Translational Symmetry	*Rotational Symmetry*	*Reflective Symmetry*	*Honest Users' Distances to Attackers*	Mobile Attacker			Stationary Attacker		
				# of Receivers			*# of Receivers*		
				1	2	3+	1	2	3+
Asymmetric	Symmetric	Symmetric	not equidistant						
			mixed				≈		
			equidistant				≈		
		Asymmetric	not equidistant						
			mixed				≈		
			equidistant				≈		
	Asymmetric	Symmetric	not equidistant				≈		
			mixed				≈	≈	
			equidistant				≈	≈	
		Asymmetric	not equidistant	≈	≈	≈	≈	≈	≈
			mixed				↓	↓	
			equidistant				↓		

Table 4: Yellow ≈ denotes no loss of privacy (beyond increased awareness of nearby phones via shared distances), red ↓ denotes a loss of privacy, and gray boxes denote combinations that cannot occur. Only a few scenarios give a privacy attacker an advantage using CoCoT over GAEN.

6.3 Increased Sensing Range from Rebroadcasting

CoCoT increases the distance at which an attacker can sense another phone. If a phone is outside an attacker's immediate sensing range (i.e., the reach of their BLE radio) but the two phones share a mutual neighbor, that neighbor will share information about the unsensed phone to the attacker, effectively increasing the attacker's area of awareness by that of the attacker's neighbors.

7 DISINFORMATION ATTACKS

An ACT attacker may wish to decrease the utility provided to users by increasing the number of false positives or negatives, causing others to conservatively waste resources or act in a way that may put more people at risk of infection, respectively. Similarly to privacy attacks, CoCoT only affects the way phones broadcast, and therefore it does not protect from GAEN's disinformation attacks, but we discuss them nonetheless (Section 7.1). CoCoT's rebroadcasting does create a novel attack surface for disinformation attacks. We discuss these and potential mitigations (Section 7.2).

7.1 Disinformation Risks Inherited from GAEN

While GAEN was designed to be secure, primarily through cryptography, some attacks still exist [17]. To trigger false-positive alerts on GAEN (and consequently CoCoT), an attacker can falsely (re)broadcast *RPI*s from infectious reporting TEKs. Guessing these is considered cryptographically infeasible but other methods exist. For example, an adversary could record *RPI*s from known-infectious users (possibly at a hospital) then rebroadcast these *RPI*s in real-time from another device in a critical area (possibly a vaccine production plant). Conversely, the best known way to trigger false negatives is to prevent users from receiving BLE chirps or uploading their TEKs when infected. For additional attacks against GAEN that CoCoT inherits, see Dehaye and Reardon's work [17].

7.2 Disinformation Risks of Rebroadcasting

Collaboration enables a new attack vector: malicious actors spreading disinformation about their environment to make risk assessments less accurate. Unlike the aforementioned attacks on GAEN, disinformation attacks on CoCoT can be done entirely locally (not requiring a remote corroborator) and can increase the rate of false negatives without noticeably denying service by carefully altering the information rebroadcast. To test the effect of this attack, we ran simulations where some phones adversarially reported disinformation to nearby phones. We tested several strategies and number of phones disreporting then evaluated the accuracy of CoCoT. We first show how CoCoT is affected by this disinformation (Section 7.2.1) then discuss detection and mitigation strategies (Section 7.2.2).

7.2.1 Attack Effectiveness. We tested three methods for how attackers might disreport information using CoCoT. Two passive strategies—under-reporting and over-reporting—that always report 0 or 15 feet, respectively, and an active strategy—*cutoff-reporting*—where adversaries wait to receive all honest broadcasts then respond with a distance calculated so the averaged distance between the

devices results in a false positive/negative using the 6 ft criteria[6]. We varied the number of devices disreporting from 1 to 16, randomly selected from the 18 total devices in the SALSA dataset [1], and measured the accuracy between honest users. We compare the percent increase in mean absolute error from CoCoT to GAEN (since GAEN is unaffected by the attack) in Figure 2. For simplicity, we do not discuss malicious sybil identities since they do not significantly strengthen the attack, see our technical report for details [31]. We show the under-reporting attack, since it decreased CoCoT's accuracy most of the three strategies.

When malicious users number fewer than 9 of 18 total, CoCoT's collaborative estimates still outperform local estimates. *Spring-Mass* algorithm's inclusion of a discrepancy term makes the algorithm less prone to disinformation attacks.

7.2.2 Detecting Disinformation. *Spring-Mass*'s *stress* can detect disinformation attacks. Distances estimated between devices will have some expected error distribution, resulting in some expected *stress* value. When adversaries disreport, *stress* deviates from its expected value, shown in Figure 2. Honest phones can define a *stress*-based threshold to detect when enough attackers are nearby to reduce CoCoT's accuracy to worse than GAEN's, at which point they stop using CoCoT. Figure 2's "Stress Thresholded" shows this strategy.

8 CoCoT PRACTICAL CONSIDERATIONS

We now enumerate potential challenges to deploying CoCoT and propose solutions. First, we discuss how devices running the CoCoT protocol will communicate (Section 8.1); then we discuss how asynchronicity affects CoCoT (Section 8.2); and, finally, we discuss CoCoT's battery consumption (Section 8.3).

8.1 BLE Communication

CoCoT, like GAEN, uses BLE data advertisements, each containing up to 31 bytes, to broadcast data to nearby phones. CoCoT broadcasts the same MAC GAEN broadcasts so devices can be identified without re-broadcasting their RPI. Additionally, we hash the RPI of each neighbor to a 3-byte digest to also avoid re-broadcasting neighbor's RPIs. Distances to neighbors are shared using 1 byte.

In total, we can describe each neighbor in 4 bytes. CoCoT tiles these descriptors into the 31-byte advertisement, describing up to 7 neighbors per broadcast. We use the remaining 3 bytes for versioning and hash-collision detection. If phones have more than 7 neighbors, CoCoT broadcasts additional advertisements.

Because RPIs are hashed, hash collisions are possible. In a dense crowd, e.g., a crowded sports arena, there can be as many as 2 people per square meter, or 0.186 people per square foot [47]. This means approximately 132 other people can be within a 15 foot radius of a phone. In this case, the probability of two nearby devices' RPIs producing the same digest is:

$$\mathbf{P}[\text{hash collision}] = 1 - \frac{2^{8\times3}P_{132}}{(2^{8\times3})^{132}} \approx 5.2\times10^{-4} \tag{8}$$

where ${}_nP_k$ is the number of k-sized permutations of n objects without replacement. The numerator is the number of ways to order 132 choices of 3 bytes without replacement and the denominator the number of ways to order them with replacement. Hash collisions are handled by using the un-hashed first 3 bytes of the higher colliding RPI instead of the hash digest. A flag in the data advertisement signals that a hash collision occurred and for which digest. The full BLE packet layout is detailed in our technical report [31].

8.2 Synchronization

We previously assumed that all phones communicated instantly, synchronously, and reliably. We now explore these assumptions.

In reality, multiple phones will sometimes broadcast simultaneously, causing packets to collide and be dropped. Broadcasting more frequently does not solve this problem, as it increases the likelihood of devices broadcasting simultaneously and colliding, especially as crowds grow. A tradeoff must be made between sending data quickly enough so other device have up-to-date information and slowly enough to minimize packet collisions.

Assume that phones broadcast at a uniform random time within the broadcast interval. The collision probability between two devices can be modeled as [30]:

$$\mathbf{P}[\textit{broadcast collision}] = 2\times\frac{\textit{broadcast time}}{\textit{broadcast interval}} \tag{9}$$

Assume these collisions are all independent. In a given time interval T, each phone will broadcast $b = T/(\textit{broadcast interval})$ times. With N phones all nearby, a phone will broadcast at least one non-colliding broadcast within time T with probability

$$\mathbf{P}[\textit{phone heard in } T] = (1-\mathbf{P}[\textit{broadcast collision}])^{Nb} \tag{10}$$

The probability N phones each transmit at least one non-colliding broadcast within time T is

$$\mathbf{P}[N \textit{ phones heard in } T] = (1-\mathbf{P}[\textit{broadcast collision}])^{2Nb} \tag{11}$$

CoCoT will broadcast up to 3 extra packets detailing the device's nearest 21 neighbors. In a dense crowd [47], at most 19 people will be in a 6 foot radius of the phone, so this will capture everyone deemed at risk by the CDC [19]. BLE can transmit one full data advertisement packet in 350 μs [6]. If phones broadcast GAEN packets every 235ms and CoCoT packets every second, 21 nearby devices will detect each other with probability ≥0.99 within 9 seconds.

[6]For example, an honest user's report of 5 ft might receive 7.1 ft in response, so the average is over 6 ft.

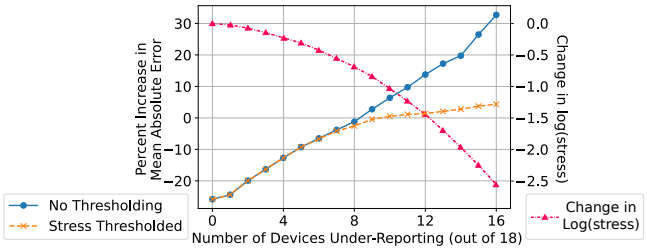


Figure 2: An increase in the number of nearby attackers decreases CoCoT's percent improvement over local estimates (left axis, higher is worse). Adversaries must control 9 of the 18 devices present to cause naïve collaboration's accuracy (blue) to be worse than local estimates. *Stress* (magenta, right axis) can be used to detect disreporting devices, allowing for counter-measures to be used (orange).

Within the 270ms that a device records a GAEN packet, estimates a distance, and rebroadcasts it, a fast walking person can travel 1.6 feet [43]. The distance traveled is made greater if the person is running. However, these interactions do not pose significant risk, especially if the users pass by each other at higher speeds [18, 50]. The interactions we most care about is when users are nearby for several minutes and relative motion is minimal between devices, making synchronization easier.

These decisions are made to be backwards compatible with BLE 4.0, a design decision also made by GAEN[7]. However, most phones built since 2017 support BLE 5.0, enabling access to extended data advertisements that transmit with twice the bit rate, have lower collision rates , and have packet sizes up to 256 bytes. Implementing CoCoT with BLE 5.0 would improve its ability to scale.

8.3 Battery Consumption

We next show the impact of CoCoT on modern devices should be unobservable by users by analyzing computation and communication cost on battery consumption.

Computation. When GAEN receives an *RPI* from a nearby device, it computes a distance estimate. Once per day, GAEN will download a list of *TEK*s of all infectious-reporting users in the health jurisdiction. GAEN hashes and encrypts each downloaded *TEK* and compares the result against every BLE-received *RPI* [23, 29]. This is unchanged by CoCoT. If a phone observes at most p phones within a 10-minute interval[8], for an average of d 10-minute intervals ($d \leq 144$), and i infected users are reported sick in the jurisdiction, then GAEN's battery consumption, C_{GAEN}, scales at worst proportionately to $O(p \times d \times i)$.

CoCoT adds computations to improve the GAEN-calculated distance estimates. These calculations are equivalent to Jacobi iterations [56, 61] on matrices no larger than 22×22 (21 neighbors + 1 self, limited by BLE 4.0); the computational complexity scales quadratically with matrix size. Hence, CoCoT's additional battery consumption, C_{CoCoT}, scales at worst proportionately to $O(p^2 \times d)$, where $p \leq 22$.

The number of infectious-reporting people, i, easily overwhelms the other terms, with 7-day averages often in the tens-of-thousands for large states like New York [11, 12]. Hence, when i is high, CoCoT's added battery consumption, C_{CoCoT}, is negligible. When i is low, the combined battery consumption $C_{CoCoT} + C_{GAEN}$ is still much lower than C_{GAEN} alone with large i. For example, with $p = 22$ (max), $d = 144$, and $i \approx 100$, which is low even for small states like Wyoming, CoCoT's battery consumption is lower than GAEN by about 5×.

Communication. GAEN's communication consists of broadcasting periodic chirps, receiving nearby users' chirps, and uploading/downloading *TEK*s from the central authority. CoCoT adds a single communication round between nearby devices, sharing recorded distance estimates. To estimate BLE power consumption, we use the Texas Instruments' BLE power consumption calculator [54]. We estimate GAEN uses 2.4mAh per day , consistent with other estimates [33]. CoCoT would add 32.4mAh in the worst case, if communicating constantly with 21 neighbors.

To contextualize this, we examine the impact on the battery consumption on an iPhone4, the oldest Apple device supporting GAEN, with a much smaller battery (1420mAh) than new devices [21]. Assuming the battery lasts for 24 hours, GAEN's BLE power usage would increase battery consumption by 0.17%[9], and CoCoT's added BLE usage would further increase it by 2.3% at worst. Following similar reasoning, on newer phones such as the iPhone 14 Plus (4325mAh) and the Pixel 7 (4355mAH) [52], CoCoT would, in the worst case, increase battery consumption by ≈ 0.75%.

Summary. Besides the computation and BLE communication discussed, GAEN performs other actions that remain unaffected by CoCoT, such as uploading and downloading *TEK*s. The impact of these on battery consumption is more difficult to calculate. However, the total battery consumption of GAEN, including on older devices with smaller batteries, is reported to be less than 5% [33].

Hence, although CoCoT requires additional BLE communication and computation, we show above that it would add only a small fraction to the battery consumption of GAEN as it is deployed now.

9 CONCLUSION

ACT has the potential to slow the spread of infectious diseases and to reduce the number of deaths caused by pandemics. While some ACT software and hardware infrastructure has now been built, much remains to be done to enable widespread adoption and realize ACT's full potential. In this paper, we demonstrate a method to improve ACT by addressing a key technical problem: making distance estimation more accurate to reduce incorrect risk assessments. Our solution, CoCoT, requires only limited additional communication and has limited impact on the privacy of the system's users while decreasing the false positive/negative rates for COVID-19 assessments by 21%. CoCoT is flexible: it provides accurate, continuous estimates which can be useful for detecting diseases that differ in transmission distances and can be readily integrated with other ACT solutions. CoCoT does not require additional infrastructure to be deployed and is compatible with those that do. It may also be useful in other application domains such as autonomous vehicles, where it could be undesirable or impossible to localize via a central authority but local ranging to nearby devices is still required.

ACKNOWLEDGEMENTS

This work was supported in part by the U.S. Army Research Office under MURI grant W911NF-21-1-0317.

REFERENCES

[1] X. Alameda-Pineda, J. Staiano, R. Subramanian, L. Batrinca, E. Ricci, B. Lepri, O. Lanz, and N. Sebe. 2016. SALSA: A Novel Dataset for Multimodal Group Behavior Analysis. IEEE PAMI (2016).
[2] J. Houchens andand J. Gold, N. Maynard, M. Krangle, and S. Kikkisetti. 2020. MITRE TC4TL Challenge System Description. NIST TC4TL Challenge (2020).
[3] S. A. Antos, M. V. Albert, and K. P. Kording. 2014. Hand, belt, pocket or bag: Practical activity tracking with mobile phones. Journal of Neuroscience Methods (2014).
[4] J. Bay, J. Kek, A. Tan, C. Sheng Hau, L. Yongquan, J. Tan, and T. Anh Quy. 2020. BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders. Government Technology Agency-Singapore, Tech. Rep (2020).

[7] BLE is the main synchronization bottleneck. Computing a round of CoCoT's updated distance estimates takes only 14ms on a Google Pixel3 with a Snapdragon 845 processor
[8] 10 minutes is used here due to GAEN's refreshing RPIs every 10 minutes.
[9] $\frac{2.4 \text{ mAh per day}}{1420 \text{ mAh per day}} \times 100\% = 0.169\%$

[5] Martin Z. Bazant and John W. M. Bush. 2020. Beyond Six Feet: A Guideline to Limit Indoor Airborne Transmission of COVID-19. MedRxiv (2020).

[6] Bluetooth SIG 2010. Core Specification. Bluetooth SIG. www.bluetooth.com/specifications/specs/core-specification-4-0/.

[7] I. Braithwaite, T. Callender, M. Bullock, and R. W. Aldridge. 2020. Automated and Partly Automated Contact Tracing: A Systematic Review to Inform the Control of COVID-19. The Lancet Digital Health (2020).

[8] U. Brandes and C. Pich. 2009. An Experimental Study on Distance-Based Graph Drawing. In International symposium on graph drawing. Springer.

[9] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi. 2018. Collaborative Sensor Network Localization: Algorithms and Practical Issues. Proc. IEEE (2018).

[10] R. Canetti, Y. T. Kalai, A. Lysyanskaya, R. L. Rivest, A. Shamir, E. Shen, A. Trachtenberg, M. Varia, and D. J. Weitzner. 2020. Privacy-Preserving Automated Exposure Notification. Cryptology ePrint.

[11] CDC. 2023. COVID Data Tracker. https://covid.cdc.gov/covid-data-tracker/#trends_currenthospitalizations_select_48,[Accessed 12-08-2023].

[12] Johns Hopkins Corona Virus Resource Center. 2023. COVID-19 Overview. https://coronavirus.jhu.edu/region. [Accessed 15-08-2023].

[13] K. Cho, Bart v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. CoRR (2014).

[14] F. Cicala, W. Wang, T. Wang, N. Li, E. Bertino, F. Liang, and Y. Yang. 2021. PURE: A Framework for Analyzing Proximity-based Contact Tracing Protocols. ACM Comput. Surv. (2021).

[15] C. Corey. 2021. PACT Datasets and Evaluation. www.mitll.github.io/PACT/datacollection.html.

[16] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. 2004. Vivaldi: A Decentralized Network Coordinate System. SIGCOMM CCR (2004).

[17] P. O. Dehaye and J. Reardon. 2020. Proximity Tracing in an Ecosystem of Surveillance Capitalism. In WPES.

[18] L. Ferretti, C. Wymant, J. Petrie, D. Tsallis, M. Kendall, A. Ledda, F. Di Lauro, A. Fowler, A. Di Francia, J. Panovska-Griffiths, L. Abeler-Dörner, M. Charalambides, M. Briers, and C. Fraser. 2023. Digital measurement of SARS-CoV-2 transmission risk from 7 million contacts. Nature (2023).

[19] Centers for Disease Control and Prevention. 2020. Public Health Guidance For Community-Related Exposure. www.cdc.gov/coronavirus/2019-ncov/php/public-health-recommendations.html.

[20] Centers for Disease Control and Prevention. 2021. CDC Updates Operational Strategy for K-12 Schools to Reflect New Evidence on Physical Distance in Classrooms. www.cdc.gov/media/releases/2021/p0319-new-evidence-classroom-physical-distance.html.

[21] W. Galan. 2010. iPhone 4 Teardown. https://www.ifixit.com/Teardown/iPhone+4+Teardown/3130.

[22] M. Girolami, F. Mavilia, and F. Delmastro. 2020. Sensing Social Interactions Through BLE Beacons and Commercial Mobile Devices. PMC (2020).

[23] Google 2022. Exposure Notifications: Helping Fight COVID-19. Google. www.google.com/covid19/exposurenotifications/.

[24] C. Gotsman and Y. Koren. 2005. Distributed graph layout for sensor networks. In International symposium on graph drawing. Springer.

[25] T. He and M. Printz. 2020. A 2-stage Classifier for Contact Detection With Bluetooth and INS Signals. NIST TC4TL Challenge (2020).

[26] R. Hinch, W. J. M. Probert, A. Nurtay, M. Kendall, C. Wymant, M. Hall, K. Lythgoe, A. B. Cruz, L. Zhao, A. Stewart, L. Ferretti, D. Montero, Jam. Warren, N. Mather, M. Abueg, N. Wu, O. Legat, K. Bentley, T. Mead, K. Van-Vuu., D. Feldner-Busztin, T. Ristori, A. Finkelstein, D. G. Bonsall, L. Abeler-Dörner, and C. Fraser. 2021. OpenABM-COVID19—An agent-based model for non-pharmaceutical interventions against COVID-19 including contact tracing. PLoS computational biology (2021).

[27] O. Holmes. 2020. Israel to Track Mobile Phones of Suspected Coronavirus Cases. www.theguardian.com/world/2020/mar/17/israel-to-track-mobile-phones-of-suspected-coronavirus-cases.

[28] J. Hong. 2021. Hong Kong to Mandate Diners Use Contact Tracing App from Dec. 9. www.bloomberg.com/news/articles/2021-11-24/hong-kong-to-mandate-diners-use-contact-tracing-app-from-dec-9.

[29] Apple Inc. 2020. Exposure Notification. www.developer.apple.com/exposure-notification/.

[30] C. Julien, C. Liu, A. L. Murphy, and G. P. Picco. 2017. BLEnd: Practical Continuous Neighbor Discovery for Bluetooth Low Energy. In 2017 16th ACM/IEEE IPSN.

[31] T. Kann, L. Bauer, and R. K. Cunningham. 2024. CoCoT: Collaborative Contact Tracing (Extended Version). Kitlhub (2024).

[32] N. Kim. 2020. "More scary than coronavirus": South Korea's health alerts expose private lives. www.theguardian.com/world/2020/mar/06/more-scary-than-coronavirus-south-koreas-health-alerts-expose-private-lives.

[33] P. H. Kindt, T. Chakraborty, and S. Chakraborty. 2021. How Reliable is Smartphone-Based Electronic Contact Tracing for COVID-19? Commun. ACM (2021).

[34] K. Kohls and C. Diaz. 2022. VerLoc: Verifiable Localization in Decentralized Systems. In USENIX Security 22.

[35] A. Kolar Rajagopal, R. Subramanian, E. Ricci, R. L. Vieriu, O. Lanz, R. Kalpathi R, and N. Sebe. 2014. Exploring transfer learning approaches for head pose classification from multi-view surveillance images. IJCV (2014).

[36] M. Krangle. 2020. MITRE Range Angle Structured. www.github.com/mkrangle/MITRE-Range-Angle-Structured.

[37] T. Li, C. Cobb, J. Yang, S. Baviskar, Y. Agarwal, B. Li, L. Bauer, and J. I. Hong. 2021. What Makes People Install a COVID-19 Contact-Tracing App? Understanding The Influence of App Design and Individual Difference on Contact-Tracing App Adoption Intention. PMC (2021).

[38] P. S. Loh. 2020. Flipping the Perspective in Contact Tracing. CoRR (2020).

[39] J. Meklenburg, M. Specter, M. Wentz, H. B., A. C., J. Cohn, G. H., L. Ivers, R. Rivest, Gerald J. Sussman, and Daniel Weitzner. 2020. SonicPACT: An Ultrasonic Ranging Method for the Private Automated Contact Tracing (PACT) Protocol. arXiv (2020).

[40] K. Merry and P. Bettinger. 2019. Smartphone GPS accuracy study in an urban environment. PloS One (2019).

[41] A. Ng. 2021. Google Promised Its Contact Tracing App Was Completely Private—But It Wasn't. https://themarkup.org/privacy/2021/04/27/google-promised-its-contact-tracing-app-was-completely-private-but-it-wasnt.

[42] NIST. 2020. NIST Pilot TC4TL Challenge. www.tc4tlchallenge.nist.gov/.

[43] T. Öberg, A. Karsznia, and K. Öberg. 1993. Basic Gait Parameters: Reference Data for Normal Subjects, 10-79 Years of Age. JRRD 30 (1993), 210–210.

[44] Association of Public Health Laboratories. [n. d.]. Exposure notifications. www.aphl.org/programs/preparedness/Crisis-Management/COVID-19-Response/Pages/exposure-notifications.aspx, [Accessed 15-08-2023].

[45] P. Howell O'Neill, T. Ryan-Mosley, and B. Johnson. 2020. A flood of coronavirus apps are tracking us. Now it's time to keep track of them. www.technologyreview.com/2020/05/07/1000961/launching-mittr-covid-tracing-tracker/.

[46] T. Parran. 1937. Shadow on the land: syphilis. AJN (1937).

[47] A. Polus, J. L. Schofer, and A. Ushpiz. 1983. Pedestrian Flow and Level of Service. Journal of transportation engineering (1983).

[48] R. L. Rivest, D.J. Weitzner, L.C. Ivers, I. Soibelman, and M.A. Zissman. 2020. PACT: Private Automated Contact Tracing. www.pact.mit.edu/.

[49] O. Sadjadi. 2020. NIST TC4TL Challenge. www.nist.gov/itl/iad/mig/nist-tc4tl-challenge.

[50] S. Samuel. 2020. Why You're Unlikely to Get the Coronavirus From Runners or Cyclists. www.vox.com/future-perfect/2020/4/24/21233226/coronavirus-runners-cyclists-airborne-infectious-dose.

[51] S. Shankar, R. Kanaparti, A. Chopra, R. Sukumaran, P. Patwa, M. Kang, A. Singh, K. P. McPherson, and R. Raskar. 2020. Proximity Sensing: Modeling and Understanding Noisy RSSI-BLE Signals and Other Mobile Sensor Data for Digital Contact Tracing. arXiv (2020).

[52] M. Spoonauer. 2022. iPhone 14 battery life tested—here's how long all four models last. https://www.tomsguide.com/news/iphone-14-battery-life-results-heres-how-long-all-four-models-last.

[53] R. Subramanian, Y. Yan, J. Staiano, O. Lanz, and N. Sebe. 2013. On the Relationship Between Head Pose, Social Attention and Personality Prediction for Unstructured and Dynamic Group Interactions. In ICMI '13.

[54] Texas Instruments 2020. Bluetooth Power Calculator Tool. Texas Instruments. www.ti.com/tool/BT-POWER-CALC.

[55] C. Troncoso, D. Bogdanov, E. Bugnion, S. Chatel, C. Cremers, S. Gürses, J. P. Hubaux, D. Jackson, J. R. Larus, W. Lueks, R. Oliveira, M. Payer, B. Preneel, A. Pyrgelis, M. Salathé, T. Stadler, and M. Veale. 2022. Deploying decentralized, privacy-preserving proximity tracing. (2022).

[56] C. F. Van Loan and G. Golub. 1996. Matrix computations (Johns Hopkins studies in mathematical sciences). Matrix Computations (1996).

[57] J. Varadarajan, R. Subramanian, S. R. Bulò, N. Ahuja, O. Lanz, and E. Ricci. 2018. Joint estimation of human pose and conversational groups from social scenes. IJCV (2018).

[58] W. Wang, Q. Chang, Q. Li, Z. Shi, and W. Chen. 2016. Indoor-Outdoor Detection Using a Smart Phone Sensor. Sensors (2016).

[59] W. F. Wells. 1934. On air-borne infection. Study II. Droplets and droplet nuclei. American Journal of Hygiene (1934).

[60] G. Zen, B. Lepri, E. Ricci, and O. Lanz. 2010. Space Speaks: Towards Socially and Personality Aware Visual Surveillance. MPVA '10.

[61] L. Zhang, L. Liu, C. Gotsman, and S. J. Gortler. 2010. An as-rigid-as-possible Approach to Sensor Network Localization. ACM TOSN (2010).

[62] Q. Zhao, H. Wen, Z. Lin, D. Xuan, and N. Shroff. 2020. On the Accuracy of Measured Proximity of Bluetooth-Based Contact Tracing Apps. In Security and Privacy in Communication Networks.

[63] Y. Zhuang, J. Yang, Y. Li, L. Qi, and N. El-Sheimy. 2016. Smartphone-based indoor localization with bluetooth low energy beacons. Sensors (2016).