Decentralized Location Consensus Through Proximity Tracing

Nathan Ahmann, Mason Chan, Alex Guan, Alan Miyazaki
Mentor: Haojian Jin

Abstract

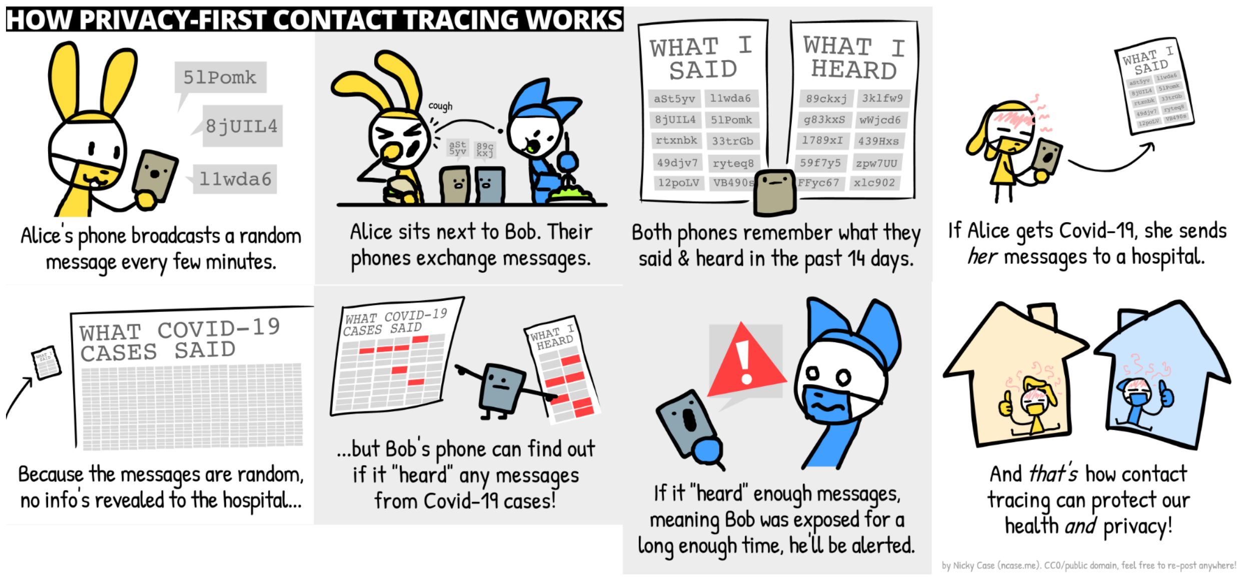
Many applications use location data to provide services, but malicious users can disrupt these by manipulating or faking their location. In an effort to combat and catch these users, our project involves a system to authenticate user location data without having one central authority in control of all the information. By leveraging the physical proximity necessary for devices to send and receive data through Bluetooth Low Energy signals, our system is able to detect users lying about their location without receiving any location information. Our portion of the project was creating the server backend that communicates with devices to store information about flagged users and process it for the purpose of adding malicious users to a blacklist.

Introduction

The project focuses on the legitimacy of mobile device locations, without compromising the privacy of its user. Several popular mobile applications revolve around proximity to other devices, such as games like Pokemon Go, dating apps like Tinder, or food apps like Grubhub. However, it can be rather easy for people to spoof or alter their location data for the sake of misleading these applications, which devalues their legitimacy and harms the user experience. Additionally, users are conscious of the fact that they want to preserve their privacy at all times, which requires applications to not use malicious software that could compromise an individual's security. Location information is considered especially sensitive and people hate feeling like they are being tracked. Due to these factors, our project aims to authenticate the location of mobile devices without violating the privacy of its users. We look towards the DP3T project as an example of this implementation, where it was able to find a strong harmony between security and location accuracy with contact tracing. Our project strives to follow the standards set by DP3T and be able to authenticate the location of our users without compromising their privacy.

Since our project is modeled after and inspired by the methods used in DP3T's COVID exposure system, it is important to thoroughly understand the work that went into their project. The goal of DP3T is to create a contact tracing method that will warn users when they were recently in contact with an individual who recently tested positive for COVID in a secure and untraceable way. Notably, the system will not keep data on positive cases, look into location hotspots for cases, or share any data about users, even for research purposes.

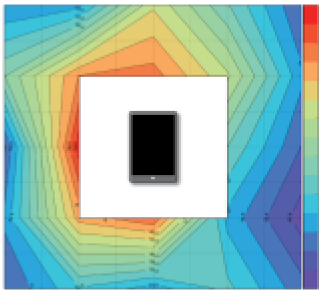
For understanding the actual methods that went into DP3T, it makes sense to start with their one-page comic summary.


HOW PRIVACY-FIRST CONTACT TRACING WORKS
51Pomk
8jUIL4
11wda6
Alice's phone broadcasts a random message every few minutes.
cough
Alice sits next to Bob. Their phones exchange messages.
WHAT I SAID
WHAT I HEARD
Both phones remember what they said & heard in the past 14 days.
If Alice gets Covid-19, she sends *her* messages to a hospital.
WHAT COVID-19 CASES SAID
Because the messages are random, no info's revealed to the hospital...
...but Bob's phone can find out if it "heard" any messages from Covid-19 cases!
If it "heard" enough messages, meaning Bob was exposed for a long enough time, he'll be alerted.
And *that's* how contact tracing can protect our health *and* privacy!
by Nicky Case (ncase.me). CC0/public domain, feel free to re-post anywhere!

The one-page comic explains the core ideas of the DP3T project. The team proposed 3 different designs of a contract tracing system and outlined the pros and cons of each in the DP3T white paper. Each design utilizes decentralized proximity tracing, where smartphones generate changing "ephemeral identifiers" (EphIDs), exchange them via Bluetooth Low Energy (BLE) beacons, and communicate with a backend server to inform other devices when an exposure event occurs.

The first design was deemed, "Low-cost decentralized proximity tracing," and as its name suggests, it focused on minimizing the data stored in servers through the use of pseudo-random timed-based EphIDs. The issue with this design is that the seed that generates EphIDs would need to be distributed to all devices when a user is exposed, which leaves an opening for traceability. To close that privacy concern, the second design, "Unlinkable decentralized proximity tracing," hashes EphIDs of positive tested users and stores them in a Cuckoo filter at the cost of more bandwidth and storage. This design also included an option for users to not submit EphIDs for certain time periods as an additional layer of privacy. Finally, the third design, "Hybrid decentralized proximity tracing," was a middle ground that combined both of the other designs. Taking from the first design, it functions by having seeds that generate EphIDs based on time, but instead of only switching seeds when the user tests positive and sends it to the backend, the seeds will instead switch every set period of time. Then only the seeds recorded during the exposure window will be uploaded and reduce the risk of traceability. Unlike the second design, these still keep IDs untraceable without requiring a cuckoo filter and related storage costs. However, unlike the first design, more seeds will need to be stored in the backend which will increase storage costs and bandwidth costs. Ultimately, this leads it to have more storage cost than the first design and less privacy than the second design, but a good balance of both.

In addition to the DP3T documentation, we additionally looked into other similar projects. This is mainly as a supplementary understanding of the methods available and because our goal is to expand this project to location tracing. Firstly, we looked into the options available for communicating between devices. DP3T chose to use BLE signals, which are a very common form of communication between devices. Bluetooth Low Energy is very similar to standard Bluetooth, but it takes less battery power to send and receive and also has a smaller data transfer limit. This is great for sending EphIDs between devices because we want them to be constantly communicated and they are small text files. The other important consideration for our project is how BLE can be used to measure the distance between devices. Typically, developers compute distance using a Received Signal Strength Indicator (RSSI) which gives a reading of how strong a received signal is. By using this value in a conversion formula, it allows them to map RSSI to distances and use that in their application. The issue is that RSSI can be inaccurate due to device and location conditions. As explained in Corona: Positioning Adjacent Device with Asymmetric Bluetooth Low Energy RSSI Distributions, signal strength can vary at different points on a device.



This asymmetric distribution means that using RSSI to calculate distance will need to take device orientations to be truly accurate. Additionally, this distribution will be different on different devices due to component location and material differences. As an alternative method, in Tracko researchers made use of both BLE and the fact that each device emits sound waves encoded with an identifier specific to that device to accurately pinpoint the relative location and distance between two devices. They chose to utilize both methods in order to get a more accurate reading of the distance between devices in their project. These sound waves are inaudible to human ears and can avoid BLE-related issues, but they have their own issues with sound not being detected in the other device or not reaching the other device's microphone.

We also looked deeper into projects that focused on using a similar system for location consensus, which is our ultimate goal. The idea of global attestation of location in mobile devices is to create a system that serves to confirm user location by corroborating their location data with other devices in the area. In “Guaranteeing the Authenticity of Location Information”, the researchers analyzed early methods of location authenticity and defined location authentication in relation to entity authentication, which is the authenticity of the information provided by other devices to verify the device in question. Additionally, that paper goes into detail about various kinds of attacks that malicious users might perform against the system. Aside from defining the types of attacks, they also provide a brief recommendation on ways to avoid a specific attack. Since the main concern with location data is privacy in order to protect users, we looked into a paper about a less theoretical security model.

Specific architecture has been created for the authentication of location as described in “VeriPlace: A Privacy-Aware Location Proof Architecture”, which details how this system can validate location data while also protecting privacy. The architecture intelligently detects suspicious activity regarding the location of a mobile device, such as the user being in multiple places at the same time. This location authentication is reliant on three main parties, where each party knows either the user’s location or identity. Separating this information ensures that one malicious party cannot breach all of another user’s information. The architecture uses two different third parties, one for location authentication and one for user identification, which work alongside a cheating detection authority to guarantee location accuracy and flag suspicious activity. The third-party for user information contains encrypted data that compares the user’s location to the other third party which contains public information. For the system to be confident in the location accuracy, the cheating detection authority compares the public location information of a user’s access point to previous ones. If two or more access points are too far apart from one another within a short period, then the authority flags the location and marks it as a sign of cheating. The paper describes how the architecture was constructed, along with how it was implemented through the example of the popular application Yelp. It also explains what the architecture considers malicious or an example of cheating, along with how each of the three parties is trusted.

Lastly, “Global attestation of location in mobile devices”, discusses a system that differs from the VeriPlace model by handling authentication on a global trust-based level rather than a local cheating level. Devices that both record each other being in the same location at the same time are said to be consistent and have their trustworthiness raised. On the other hand, devices that make conflicting claims of having seen or not seen each other have their trustworthiness lowered. Then these trustworthiness scores are converted into matrices and used to compute a global trust value via the EigenTrust and PeerTrust models. When determining if a device truly is where it claims, the system finds devices that claim to be near the device and devices that claim to be in the area regardless of if they have seen the device in question. After sorting the available devices by trustworthiness, the claim from the device with the highest score is taken as the consensus report and used as the truth.

Methods

Our project sought to identify a method in which we could use Bluetooth signals from mobile devices to verify user location without creating a privacy concern. To do so, we initially modeled our solution off of the base code structure of the D3PT project. After several roadblocks in getting their code to properly run and adapting it to our needs, we ended up making our own backend from scratch. We now had the issue of trying to figure out what we would need to create this whole service. For our server, we connected Heroku, Postgres, and Django to create a website functioning as a backend database. This would allow us to begin storing some test data which would lead the way to begin to store messages from phones and send replies back.

Once we had the tools established, we needed to figure out just how we would create an app that uses location data while not compromising privacy. Our solution is to never send the location data to a server in the first place. Location data will all be handled using Bluetooth between phones and the only thing the phones will send to the server will just be the ID of the phones that are in the same location as them physically, yet claim to be in a different location. More specifically, users can fake their location by using a false GPS location, but their physical location can't be faked. To accomplish this, we leveraged BLE signals, which can only be sent between devices physically near each other. Phones near each other will communicate with each other and identify which phones claim to be at a location different than their own. If a BLE signal claims to be at a different location than a user, one of the two devices is lying about its location. Phones will collect lists of IDs for devices they detect with conflicting locations and then send that to the server. This allows the server to collect the IDs of users claiming to be at different locations without ever sending location information to the server. While location information is collected on the phones, it is not stored and is only used locally and any phone close enough to receive a BLE signal would already know your location since it would have to be within a few meters.

After the data is collected, it still needs to be processed on the server. We cannot just mark any ID received as a fraudulent user and instead have to create a trust model to determine which users are lying. For example, consider a situation where users A, B, and C are all at the same location, yet C is lying about their location. Our server would receive claims from users A and B that user C is lying, but we would also receive a claim from user C that both A and B are lying. To resolve these claims, we need some form of a trust algorithm that can systematically determine which users should be added to the blacklist. We opted to go with one of the most simple trust algorithms: a majority vote. Based on the data it receives, our server adds users to a blacklist if at least 50% of users claim they are lying within the time window. Despite its simplicity, this algorithm can effectively identify lying users and operates under a fair outlook that everyone is telling the truth. While more sophisticated methods could improve the logic, address edge cases, or provide better defense against foreseeable attacks, our goal was to create a functioning system with our limited knowledge of cybersecurity and backend design. With those limitations in mind, this option provided the best balance of privacy and complexity.

Results/Discussion

Our project successfully created a system that can identify users lying about their location data, while also keeping their information private and decentralized. Utilizing the physical proximity required for devices to send data through Bluetooth Low Energy allows us a way to track a user's true location through other users instead of relying on sending the data to and from a system. Additionally, we created a fair protocol that assumes all users are being truthful and only seeks to identify lying users. This fair protocol uses majority voting and adds users who are flagged by a majority of users to a blacklist that can then be used by any authentication service to check if a user is likely faking their location. This operates under the idea that most users do not attempt to fake their location and that those who do are repeat offenders and will attempt to lie multiple times.

All that has been discussed so far is about the current state of our project and there is room for improvement and future growth. Our goal was to create a system meeting all of our requirements and as many of our hopes as possible while taking into account our knowledge and skills. This means that there are several limitations and possible solutions to address. Notably, our system requires at least 3 devices at any given location in order for 2 truthful users to have a majority over a lying user. Additionally, our current system has issues dealing with multiple groups of users that create non-connected interaction sets. Both of these were fine under our goal of creating this system for use over the UCSD campus, but would not hold up if implemented at a large scale. Potential ways to address these problems would involve more knowledge of cybersecurity or more sophisticated trust algorithms. The second issue of multiple user groups could be addressed by sending data of all users seen by a device instead of just IDs of devices sending conflicting location data, but that would encroach on user privacy. By utilizing data on all devices a user came into contact with, one could create a web of social interaction or possibly identify user location by mapping their paths with other users until it reaches a user with a known location. Since user privacy was at the forefront of our project, we opted to not pursue this route.

Lastly, concerning cybersecurity, our group members had little knowledge of attacks and defenses against them. For testing purposes and to ensure a smooth connection between our server side of the project and the app side, we left our server very open. If these ideas were to be deployed into any actual environment we would suggest taking proper precautions to lock down the server data, even though it only contains user IDs which are randomly generated hashes. Following privacy techniques present in D3PT could be even better, since they implemented a system that changes IDs over time to avoid exposed data being able to be linked to any user.

References

Arunkumar, S., Srivatsa, M., Sensoy, M., & Rajarajan, M. (2015). Global attestation of location in Mobile devices. *MILCOM 2015 - 2015 IEEE Military Communications Conference*. https://doi.org/10.1109/milcom.2015.7357675

DP-3T. (n.d.). *DP-3T/documents*. GitHub. Retrieved 2022, from https://github.com/DP-3T/documents/tree/master/public_engagement/cartoon

Ferreres, A. I., Álvarez, B. R., & Garnacho, A. R. (2008). Guaranteeing the authenticity of location information. *IEEE Pervasive Computing*, *7*(3), 72–80. https://doi.org/10.1109/mprv.2008.49

Jin, H., Holz, C., & Hornbæk, K. (2015). Tracko. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. https://doi.org/10.1145/2807442.2807475

Jin, H., Xu, C., & Lyons, K. (2015). Corona. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. https://doi.org/10.1145/2807442.2807485

Luo, W., & Hengartner, U. (2010). Veriplace. *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. https://doi.org/10.1145/1869790.1869797