ABSTRACT

Location-aware mobile applications are rapidly gaining popularity. This growth has caused the emergence of services that are offered to the users only when they are at specific locations. To implement valuable services, like a product sale, it is necessary to verify the presence of the user’s device in a way which can be reliably trusted by the providers.

We present SureThing, a location proof solution for mobile services that need assurances about the physical location of a user’s device. Its goal is to support the creation of proofs which offer evidence that the user’s device is at a claimed location. SureThing relies on different techniques for location estimation and on other devices that can testify to the presence of the user’s device and to the credibility of the location measurements.

A SureThing prototype was implemented in the Android platform. The prototype was evaluated regarding response times, accuracy of location estimations, and feasibility of proof exchanges between user devices. The results show that the solution is both practical and useful.

CCS CONCEPTS

- Human-centered computing → Computer supported cooperative work; Ubiquitous and mobile computing systems and tools;

KEYWORDS

Mobile Security, Context-Awareness, Location Estimation, Location Proof, Internet of Things

1 INTRODUCTION

The use of multiple sensors and actuators, embedded in the environment and connected to computers, enables context-aware systems that gather information about the real world for use in virtual systems. In fact, the main goal of the Internet of Things (IoT) [15] is to connect and integrate the real and virtual world to allow new and useful applications. The smartphone is the device that can better take advantage of this environment awareness enabled by the IoT because of its numerous sensors, Internet connection and, most important, because of its close bond with a human user.

Location is one of the most used types of context [3] in smartphone applications. By observing the location of a user, it is possible to provide a personalized service to her [6]. Each point of interest in the physical world can be labeled to provide information or to trigger actions in an application. For example, consider a service that provides access to products in a vending machine. If the user’s device is certifiably at a particular location, at a given time, and the payment was made, then permission to retrieve the product can be granted. Trusted attributes are required to make these kinds of secure authorization decisions. A location claim made by the user is not enough to give an intended privilege. Location data needs to be collected and verified so that service providers can have strong assurances about the claimed location.

We present SureThing, a location proof solution for mobile devices, which lets applications abstract themselves from location estimation and proofing mechanisms. SureThing uses multiple location estimation techniques and relies on witnesses to testify the presence of a user at a given place [10, 16]. These witnesses are other users of the system, carrying their smartphones running the application, that happen to be in the same location as the user. A SureThing prototype was implemented and evaluated in the Android platform.

Motivating Use Case

The following scenario illustrates the practical use of SureThing. Consider a state-of-the-art shopping center that wants to improve its customers experience by rewarding regular visitors with special discounts. Alice is a customer that opted-in to the program and is running the loyalty application on her smartphone. Alice arrives at the food court of the shopping center and is deciding where to have dinner. Her presence is detected and her smartphone is asked to prove her location. The history of Alice’s presences at the location is also retrieved. Alice’s device collects location measurements to make a location claim and, at the same time, requests a location proof to a nearby witness. Bob, another user running the application on his smartphone, was also at the food court and was selected as witness. Alice presents her claim and the proof provided by Bob to the shopping center. The proof is verified, everything is OK, and as a result, Alice receives discounts for restaurants. She can now happily go to one of her favorite places at a very special price. The shopping center has just rewarded a verified regular customer that will, hopefully, come back more often.

SureThing was used by the loyalty application to abstract the used location estimation technique and to manage the cooperation between users to provide the witness proofs.
Document Overview

The rest of this paper is organized as follows. Section 2 presents location estimation techniques that will be useful in the development of SureThing. Section 3 presents the most relevant techniques for location proofing that were used in previous works. Section 4 details our location proofing solution. Section 5 presents the evaluation conducted to validate the design and implementation of the system. Finally, Section 6 presents conclusions and future work.

2 BACKGROUND

There are alternative ways for a user to measure her location. GPS [12] is highly available but it does not work indoors. Wireless networks can be used as an alternative to GPS, using cell towers [9] or Wi-Fi access points [5, 8]. Wi-Fi location systems use a location fingerprint, a “map” of signal strengths, obtained by receiving signals from the different wireless access points. It is then possible to measure the signal strengths in the user’s device and compare these measurements with the values that are stored in the map. The main disadvantage of this method is the fluctuation of signal strengths that can produce estimation errors.

The Android Network Location Provider (ANLP)\(^1\) is a practical network-based location system that uses both cell tower and Wi-Fi access point information in order to better determine the location of a user. This method responds faster and uses less battery than GPS solutions and achieves more precise results in areas with more Wi-Fi access points.

Bluetooth location systems are based on beacons spread over the area to be covered [1]. Since Bluetooth range is limited to typically less than 10 meters, it is assumed that if a user can detect a beacon, then she is near the location. However, to cover a large area, many beacons need to be spread over the area which imposes high hardware and installation costs. In the last few years, a new version of Bluetooth known as Bluetooth Low Energy (BLE) has emerged. BLE is mainly used in situations where the amount of data to be transferred is low. It is also more power efficient than the original Bluetooth [7]. The iBeacon protocol [11] is an example of the use of BLE to determine the location of a user. The key device of this protocol is a BLE beacon. This device is constantly emitting values that can be used by the applications to derive the place where the user is, if the values were previously associated with logical places.

QR codes\(^2\) can also be used to develop location aware systems. However, location estimation techniques based on QR codes assume that the user has to scan the code, while other solutions provide methods where the location estimation does not need the participation of the user. As in Bluetooth, we also have to install further infrastructure in the places we want to identify.

Another approach to determine location is Ambient Fingerprinting [2]. The smartphone can sense, for example, light and sound, and this ambient data makes it possible to construct a fingerprint for each place. However, this ambient “map” needs to account for variations from day to day or even from hour to hour, which makes it more difficult to maintain a well populated fingerprint.

All location estimation techniques presented have some strong and weak points as previously described, and can adapt differently to distinct situations.

3 RELATED WORK

Regarding location proofing, the main concept is a location proof, defined by Saroiu et al. [14], as a piece of information that ensures that a user is at a given place, at a given time. If the user presents a valid proof to an external service, the latter can have more certainty about the verity of the user’s claim. There are systems built on this location proof definition and that use witnesses in order to verify the presence of a user in a given space.

The APPLAUS system [17] works without depending on any infrastructure from the place where a proof is generated. By using a peer-to-peer approach between devices, a user only has to find someone who can give him a location proof. The user requesting for a proof is known as Prover and the entity that creates a location proof is known as Witness. The only step of the process where network connection is needed is when the location proof is sent to a remote database. The service provider, known as Verifier, can later retrieve the proofs from the database. There is also a trusted Certification Authority responsible for generating public key certificates for users. APPLAUS has a collusion detection mechanism. If two users say that they are at a given location when they are not, the server checks if there are other proofs generated from the same place by any other witnesses. If there are, but those witnesses did not generate any proof to the requesting users, the latter are assumed to be possible colluders.

The CREPUSCOLO system [4] also uses a peer-to-peer approach but it depends on the installation of new infrastructure, known as Token Providers. These entities act like fixed witnesses and they also testify the presence of a user, endorsing the proof provided to the Verifier. However, if the user is not detected by the Token Provider, this will not add any improvement to the system.

Khan et al. [10] introduced a Location Authority that is responsible for selecting witnesses that should provide the proof. This makes communication between users indirect.
and collusion less likely [10]. However, the installation of added infrastructures can be an obstacle for the creation of new location-based applications as it requires up-front investment.

4 SURETHING DESIGN

Mobile application developers and service providers will find in SureThing the possibility to ask for location proofs based on different evidences, and with different degrees of assurance. SureThing uses different technologies, such as GPS, Wi-Fi or Bluetooth in order to understand where the user is located. Our system also relies on the definition of witnesses [4, 10, 16, 17] in order to help proving the location of a given user. These witnesses are other users of the system, carrying their smartphone that is running SureThing’s application. Witnesses are in the same location as the user who needs a location proof. One of the main advantages of this witness-based approach is the independence from the infrastructure placed at a given space. It allows to create location proofs by using the community present at a given place. Two of the most important challenges in this approach are what to do when there are no witnesses available nearby to certify the presence of a user and how to deal with collusion attacks. We assume that smartphone users are willing to share their locations, network bandwidth and battery.

Roles

There are four entities in SureThing (similar to the ones defined in other witness-based systems as APPLAUS [17] and CREPUSCOLO [4]):

- **Prover** - entity that needs to prove its location and gathers location proofs from its neighbors;
- **Witness** - entity that agrees to give a location proof to the Prover;
- **Verifier** - entity that specifies which location estimation techniques to use and to whom the Prover sends the proof once it receives it from the Witness. The Verifier checks if the proof is valid and informs the Prover. The Verifier is also the reference clock;
- **Certification Authority** - entity that certifies public keys and that is assumed to be trustful.

Choreography

Figure 1 presents how the SureThing entities communicate when a location proof is requested. In step 1 the Prover asks to the Verifier how it should obtain a location proof. The Verifier answers with a **Proof Demand** in step 2. This demand specifies how the Prover and the Witness should obtain their location. The Verifier also informs the Prover about what type of witnesses (described in the following of this Section) it should contact. The Prover sends a **Proof Request** to the witnesses nearby in step 3. This request contains the identification of the Prover and the demand previously received. In step 4 the Witness returns a proof to the Prover. This proof is signed with the private key of the Witness. In step 5 the Prover simply forwards the proof to the Verifier. Step 6 and 7 indicate the interaction between the Verifier and Certification Authority. The Verifier needs to check the signature in the proof so it requests to the Certification Authority the public key certificate of the Witness. After evaluating the proof, the Verifier decides to accept or reject it. In step 8, it informs the Prover about this decision. If the proof was accepted, the Prover can now receive his privilege.

![Figure 1: Communication between entities in SureThing.](image_url)

Location Proof Techniques

A location proof is a piece of information that can attest that a user is at a given place. A proof contains the following six attributes:

- **Prover ID** - identifier of the entity who needs a proof;
- **Witness ID** - identifier of the entity who testifies the presence of the Prover in a given place;
- **Location of the Prover** - location data of the Prover obtained through his smartphone sensors;
- **Location of the Witness** - location data of the Witness obtained through his smartphone sensors;
- **Nonce** - arbitrary number that was previously sent by the Verifier. It will ensure that old proofs cannot be reused;
- **Signature** - digital signature of the proof done by the Witness to guarantee content integrity and authenticity.
Both the Prover and the Witness have to collect location data so that the Verifier can compare the information given by the two and check if it yields the same place.

Three different proof techniques were developed, as shown in Table 1: Geo, Wi-Fi and Beacon. Ambient Fingerprinting [2] was not included in the current prototype because wireless measurements are more widely available and also because the use of camera and microphone raised bigger privacy concerns.

Each of the proof techniques requires a setup stage to be performed by the service provider before location proofs can be made. SureThing includes the necessary tools for the setup e.g. tools to populate Wi-Fi fingerprints. The setup includes adding logical descriptions of the physical places. For example, in the shopping center scenario, it would be up to the center staff to perform the setup procedures for each technique to be used and to label the places adequately e.g. food court, cinema, etc.

In the Geo proofs, different geographic location measurements are collected from inside the area to identify. Each location is defined by geographical coordinates, as latitude and longitude, and also has an associated radius. With that information, the area is divided in multiple circles and the Verifier can then check if the locations given by Prover and Witness are inside the same area.

In Wi-Fi proofs, multiple readings are done to construct a fingerprint: list of access points with an associated signal strength value. Each zone will have its own fingerprint.

For the Beacon proofs, a specific implementation of the iBeacon standard [11] was used. These beacons are always emitting a value via Bluetooth and in the setup phase each beacon value is associated with the corresponding place.

The Verifier always has to perform two tasks, regardless of the proof technique being used. First, it has to perform the verification of the proof’s digital signature, made by the Witness. Second, the Verifier must check if the nonce received in the Proof is the same that it previously sent to the Prover, when the latter requested a proof demand. Table 2 presents the remaining tasks done by the Verifier to verify each type of proof.

**Witness Models**

When a Prover needs a location proof, it will request it from one or more devices nearby, known as witnesses. Previous solutions that used a witness-based approach focused on random mobile witnesses. Usually, in systems that use random mobile witnesses, users can behave as Provers or Witnesses for other Provers. We envision that, although random mobile witnesses are a good solution, since they allow to gather proofs from close peers without any installment of further infrastructure, they will not fit in some use cases for location-aware applications. For example, if a Prover \( P \) is at a place with low attendance, it is possible that a Witness of this type will never generate him a proof, since \( P \) will never find a Witness. We also envision that in some cases we can differentiate between trusted and untrusted witnesses.

SureThing offers two main types of witness models: Master and Mobile. If no witnesses are available nearby, the service provider has the opportunity to ask for a secondary witness model, known as Self Witness model, that will generate a weak proof. We present our witness models below:

- **Master** - certified witness that can be trusted by the Verifier e.g. an employee of the shopping center.
- **Mobile** - untrusted random witness e.g. a client of the shopping center. The Verifier does not trust the proofs gave by mobile witnesses because they can be colluding with \( P \);
- **Self** - \( P \) acts like his own witness and generates a proof for himself.

<table>
<thead>
<tr>
<th>Proof Technique</th>
<th>Location Data Included in the Proof</th>
<th>Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo Proof</td>
<td>Geographic Location (obtained from GPS or ANLP)</td>
<td>Collect geographic coordinates with a correlated radius and associate them with their corresponding logical place</td>
</tr>
<tr>
<td>Wi-Fi Proof</td>
<td>Wi-Fi Fingerprint</td>
<td>Collect Wi-Fi fingerprints for each place and associated them with their corresponding logical place</td>
</tr>
<tr>
<td>Beacon Proof</td>
<td>Closest beacon ID detected</td>
<td>Associate beacon values with their corresponding logical place</td>
</tr>
</tbody>
</table>

Table 1: Proof Techniques Fields and Setup Phase.

---

3Estimote beacons were used. More information at [https://estimote.com/](https://estimote.com/)
<table>
<thead>
<tr>
<th>Proof Technique</th>
<th>Verifier Tasks performed upon receiving Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo Proof</td>
<td>1. Verify if geographic locations of the Prover and Witness are close to each other (threshold is configurable); 2. Check if the location of the Prover and Witness are inside any of the previously saved areas; 3. If both are inside the same area, the Verifier accepts the proof. Otherwise, it rejects it.</td>
</tr>
<tr>
<td>Wi-Fi Proof</td>
<td>1. Get places with closest fingerprint to the one given by the Prover and Witness; 2. Compare places. If they are different, rejects proof; 3. Verify if fingerprints of the Prover and Witness have a minimum amount of access points in common with the saved fingerprint for that place. If they have, the proof is accepted. Otherwise, it is rejected.</td>
</tr>
<tr>
<td>Beacon Proof</td>
<td>1. Compare beacon values provided by the Prover and Witness; 2. If they are the same and there is a place associated with that beacon value, accepts proof. Otherwise, the Verifier rejects it.</td>
</tr>
</tbody>
</table>

Table 2: Verifier Tasks for each Proof Technique.

Implementation

We implemented a SureThing prototype for mobile devices running Android OS, the most widely used mobile operative system. Bluetooth was chosen for communication between mobile devices, mainly due to its short range. It is also a widely used technology, present in almost every smartphone model. Normally, the Bluetooth specifications require a pairing process between devices before any data is transferred. However, this specification is not a good fit for the purposes of SureThing. In most cases, Prover and Witness will not know each other, so they cannot share any information to pair their devices. We also want our solution to have the least user intervention possible. We use Bluetooth communications without pairing. The potential security implications of this option are addressed in the application layer. To guarantee non-repudiation and integrity of the proofs, witnesses have to sign them with their own private key. Witnesses are always listening to connection requests. Those connections are initiated by the Prover when it needs a proof.

Both the Verifier and Certification Authority are implemented as RESTful web services written in Java. The data transferred to these two entities and from them is in JSON, a text-based data exchange format, that is less verbose than XML but is still convenient for use in object-oriented programming languages.

Collusion Avoidance Mechanisms

One of the problems that arise from solutions with random mobile witnesses is the possibility of collusion between Prover and Witness. If both Alice and Bob lie, the Verifier must have mechanisms to detect such malicious acts. In the current prototype we started with Witness Redundancy and then introduced an improved mechanism called Witness Decay.

In the Witness Redundancy mechanism, the Prover has to gather proofs from multiple Witnesses instead of only one. This number should vary with the value of the service that is being offered and with the level of attendance of the place where the proof is going to be generated.

In the Witness Decay mechanism, each proof has a different value associated with it. If a Prover is always getting proofs from the same Witness, its proofs gradually become less valuable. If the Prover cannot gather proofs with enough value to satisfy the Verifier, the latter will not provide any service to the Prover.

The proof value is calculated with Formula 1 where $V_{xy}$ represents the proof value given to user $x$ by Witness $y$ and $V$ represents the maximum proof value. $N_{xy}$ represents the number of times that $y$ testified the presence of $x$ and $U$ is the total number of users in the place. If the Witness never testified the presence of the Prover ($N_{xy}=0$), the generated proof will have the maximum possible value. New proofs created by the same Witness to the same Prover will gradually have less value ($N_{xy} > 1$). The probability of finding the same Witness again is given by $\frac{1}{U}$. We multiply this probability by the number of times that the Witness gave a proof to the Prover, in order to penalize proofs where Witnesses and Prover are repeated. The parameter $k$ can be specified by the application to increase or decrease the speed of the Witness decay. After calculating this decay factor, we subtract it from $V$. 

$$V_{xy} = \begin{cases} 
V & \text{if } N_{xy} = 0 \\
V - \frac{N_{xy}k}{U} & \text{if } N_{xy} > 1 
\end{cases}$$  

(1)
the maximum proof value possible, in order to obtain the final value of the proof.

5 EVALUATION

In this Section we present and discuss the evaluation results of the SureThing prototype. We focused the evaluation on three different aspects:

- How accurate are the location estimation techniques?
- How long does it take to prove a location?
- Are the collusion avoidance mechanisms suitable to real world spaces?

Location estimation accuracy

As a scenario for evaluation, we used a real world building to stand in as the shopping center. The location map is represented in Figure 2(a). We divided the area in five main zones: Cinema, Food, Clothing, Technology and Children. Figure 2(b) represents the areas that were defined for use in Geo proofs, one for each zone. We tested the location estimation accuracy for each technique proposed, starting with Geo and Wi-Fi.

Geo and Wi-Fi. Since Geo proofs are used in an indoor environment, we chose to obtain geographic coordinates from the ANLP. Each of the shopping zones has its own fingerprint that was collected after 100 signal strength readings, to ensure that readings with poor values did not have a strong impact in the final Wi-Fi fingerprint.

For Geo and Wi-Fi proofs, our goal was to identify in which area from Figure 2(a) the user is in. We tested Wi-Fi fingerprinting to determine the current shopping zone where the user is, with different quantities of readings. In each reading, all the access points found were saved with an associated signal strength value. We wanted to compare if, by doing multiple readings, we could diminish the positioning error. For both techniques, we did 30 estimates per area. Our results are presented in Figure 3. We can observe that, for the majority of places inside the shopping center, Wi-Fi fingerprinting with 10 readings was the technique with the highest score. By doing just 1 reading, it was not possible to correct errors, since not enough access points to calculate a good fingerprint were found. When we increased the number of readings to 5, we had an increase in location accuracy, since it is possible to detect more access points and if some reading was poor, the others can correct it. With 10 readings, a poor reading will not affect as much the final result, leading to better results. We did not increase any further the number of readings because each reading took about 550ms. If we continued to increase this number, the location proof will take too long to be generated. Geo has, most of the time, a lower accuracy than Wi-Fi fingerprint (with 5 and 10 readings).

We also tested if we could detect if a user is not in any of the building areas. As Figure 3 shows, Geo was able to identify half of the situations. Wi-Fi fingerprint had a success rate over 80% when the user was outside of the building.

Beacon. For Beacon proofs, we had three Estimote beacons available. We wanted to evaluate if the beacons would have enough location precision to identify smaller areas inside the different zones presented in Figure 2(a).

To obtain the location estimation accuracy with beacons, we prepared a scenario as in Figure 4. Each beacon is 5 meters away from the previous one and in a different room. Tests were done with the user standing at 1 meter from the beacon. Table 3 summarizes the results. We made 50 location estimations near each beacon. The beacon in the Vegan restaurant has the lowest accuracy since it was in the

![Figure 2: Shopping Center building.](image)

![Figure 3: Location accuracy for Geo and Wi-Fi.](image)
middle of the two other beacons. We got a success rate of approximately 80% after the analysis of the three beacons. This means that beacons will provide a correct location 80% of the times when separated by 5 meters.

### Proof time

We compared the three techniques regarding the time that each one took to complete. These measures were obtained after 30 proof generations.

Figure 5 represents the time that was needed to obtain the location of the user. We can observe that geographic coordinates were the fastest to be collected. However, the mean value represented for Geo proofs is far from the median value. Half of the measurements were done below 1.61 seconds. The average and median are apart because we defined that an obtained geographic location must have an accuracy lower than a threshold defined to be 20 meters, given the shape of the building in the experiment. This threshold should be adapted for the specific application. In very large areas, a high threshold would be acceptable, because the user is probably still inside the area. In smaller areas, low thresholds have to be used. It may sometimes take longer to achieve this accuracy, which increases the average time of this proof technique, but not the median. The Wi-Fi fingerprint location technique is the one with the most regular readings regarding time spent. Mean and median values were practically the same. In Beacon proofs, a Beacon Manager is used to interact with the beacons and it can take sometimes longer to connect or to discover the closest beacon, which increases the time needed to obtain the location data. In the Wi-Fi proof, we did 10 readings to obtain a fingerprint, since it was the Wi-Fi fingerprint measure with higher location estimation accuracy. For the beacon value we used the first value read by the Beacon Manager, as recommended by the technical guidelines of Estimote Beacons.

Figure 6 demonstrates the total time, from the moment when the Prover asks for a proof demand to the Verifier, to the moment when the Verifier informs the Prover about the rejection or acceptance of the proof in the Master and Mobile Witness model. We only have one witness in our setup when testing each model, which will result in similar time measurements for both models. However, in real world scenarios is expected that a Master Witness can take a little bit longer to be found, since the Prover is looking for a specific subset of witnesses. We can see that higher the location accuracy, the more time will be needed to generate a proof. Most of the time is spent on obtaining location data and this is what differentiates the total time between techniques. For example, in the Wi-Fi fingerprint, 10.76ms are spent in obtaining location information (5.38ms in each device). Each technique spends approximately between 4.5 and 5 seconds in processes other than obtaining location data. These main processes are: the signing of the proof (112ms), the verification of the proof by the Verifier (85ms, approximately the same time regardless of the used location technique), and the establishment of Bluetooth connections between devices and communication between entities. The time overhead in the Beacon proof is also caused due to the initialization of the Beacon Manager that took, on average, 750ms in each device.

---

### Table 3: Location estimation accuracy with beacons.

<table>
<thead>
<tr>
<th></th>
<th>Sushi</th>
<th>Vegan</th>
<th>Pizza</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Claims</td>
<td>85%</td>
<td>72%</td>
<td>81%</td>
</tr>
<tr>
<td>Wrong Claims</td>
<td>15%</td>
<td>28%</td>
<td>19%</td>
</tr>
</tbody>
</table>

---

5 Location Object in Android: https://developer.android.com/reference/android/location/Location.html

5 Beacon Manager: https://estimote.github.io/Android-SDK/JavaDocs/com/estimote/sdk/BeaconManager.html
The Self Witness model is a last resort model that the Verifier can use when is offering services with less value and when there are scenarios where is probable that no witnesses are found by the Prover. In this model there is no communication between Prover and another Witness, since the first will act as its own Witness. This reduces the time needed to generate a proof, since there will be only one location estimation and there are no communications via Bluetooth. The mean time spent for each proof technique was close to half of the time spent in the Master and Mobile Witness model.

Comparison of proof techniques

We evaluated three different types of location estimation: geographic coordinates, by using the ANLP; a grid of Wi-Fi fingerprints constructed by the application developer; and the use of Bluetooth beacons spread over the area. As expected, the beacons gave the highest accuracy, even if measuring smaller areas. Wi-Fi fingerprint with 10 readings showed the ability to distinguish between areas more precisely than geographic coordinates, but it requires a more careful setup phase, where a grid of Wi-Fi fingerprints has to be constructed. Our Wi-Fi fingerprint map can be divided as the application developer wants. We showed an evaluation test in which we try to divide the shopping center in five areas to compare its results against the geographic coordinates provided by the ANLP. Our Wi-Fi fingerprint location estimation accuracy is above 70% for our evaluation setup, but it can be used in larger or smaller areas. However, its accuracy should decrease when the areas to identify are smaller.

Regarding proof time, the more accurate the location technique is, the more time is needed to generate a proof. One time constraint that can affect the feasibility of the system is the time that Prover and Witness are in the Bluetooth range of each other. If the Witness takes too long to generate a proof, the Prover can now be too far to receive it. We think that in the shopping center example, our time measurements are enough to claim the feasibility of our system. For example, in a food court users tend to walk slowly or be sitting, which increases the probability of Prover and Witness staying in range of each other.

In our tests, we only used one Witness. However, the total time should not increase linearly with the number of Witnesses that received a proof request since multiple witnesses can be contacted in parallel.

Given the accuracy that our location estimation techniques achieve, we propose that the three of them can be used simultaneously. In our example, the ANLP (Geo Proofs) can be used to detect when the user enters the shopping center. It does not indicate an exact zone of the shopping but by knowing that the user is in the shopping, the other two location estimation techniques can be activated. If the Wi-Fi fingerprint indicates that the user is in a zone with Bluetooth beacons, the user can try to detect the closer beacon, so that the system can infer the store where the user is. The location estimation techniques are complementary to one another and can be used together.

Collusion Avoidance Simulation

We wanted to know that in a real environment our collusion avoidance mechanisms would not compromise the normal behavior of our system. We simulated the shopping center using Netlogo, a multi-agent programmable simulating environment. Figure 7 presents a screen capture of our simulation environment. Grey areas represent the stores, white areas represent the corridors, black represents walls and the blue dots are the users. We used this simulation to see how many Witnesses a user finds each time it requests a proof. We also studied the probability of a Prover finding the same Witness multiple times, which will cause a decay in the value of the proof.

Simulation setup. Our environment is represented as a grid with two coordinates. The horizontal coordinate goes

6Netlogo: https://ccl.northwestern.edu/netlogo/
from 0 to 255 and the vertical goes from 0 to 127. Each cell can only be occupied by one user. There are 250 users in our simulation. Netlogo provides a tick counter, that is equivalent to a time counter. Users may only move one cell per tick. Users move randomly around the shopping center. However, while in a corridor, they have a 70% probability of moving forward in each tick and only a 15% probability of changing direction. Inside a store, the moving probability decreases to 50% and the rotating one increases to 40%. Although it is not an exact model of how people walk in a shopping center, these probabilities help to decrease the randomness in the movement of the users.

At each tick of the simulation, each user has a 1% probability of requesting a proof. The Witnesses can be at a maximum distance of 10 cells from the requesting user. After 10 ticks, the Witnesses that remained nearby the Prover will respond to him. This simulates the time that Prover and Witness take when changing data via Bluetooth. We used Formula 1 defined in Section 4 to determine the value of proofs given by the Witnesses. The parameter $k$ was defined as 3 so we can have a fast decay. We defined the maximum value of each proof as 1. In this simulation, the Verifier accepts batches of proofs with at least value of 1, which means that if a Prover receives a proof from a Witness never before seen, the Verifier will accept the claim. A proof batch is a group of proofs provided by the Witnesses that the Prover contacted.

Simulation results. We ran the experiment with three breakpoints: after the request of 50, 100 and 500 proof batches. Each proof batch is generated when the Prover receives a group of proofs from Witnesses. We analyzed these different breakpoints so that we can observe how the system behaves in different situations. For example, after 500 proof batches are generated, it is expected that the provers are starting to find more repeated Witnesses than after the generation of only 100 proof batches. The number of cases where no Witnesses are found remains constant and near to 20%. In the other cases, proof batches are accepted or rejected. Results are shown in Figure 8. The number of proofs accepted decreases when there are more requests. This is caused by the Witness Decay since after a while Witnesses start to be repeated and their value decreases as defined in Equation 1. There are also no changes in the population, since the 250 users are always the same. In a real-world environment we can find more dynamic systems, which will benefit our solution. We also analyzed the average number of Witnesses found per request. With our described setup, with 250 users, there was an average of only 2 Witnesses found. If we increased the population to 1000 users, the number of Witnesses increases to 7. This demonstrates that depending on the number of users in the space, the Verifier can request proofs from a different number of Witnesses.

If a user wants to deceive the system, he will have to collude with false Witnesses. If the Verifier asks for proofs from N different Witnesses (Witness Redundancy), the user will have to gather proofs from N dishonest Witnesses. As we have shown before, in a setup with 250 users, provers usually only find 2 Witnesses, which may not be too difficult for the malicious user to gather. Our Witness Decay mechanism guarantees that the dishonest Prover cannot re-use the same Witnesses too much. Figure 9 shows the quantity of Proof Batches that are accepted when a malicious Prover is always getting proofs from the same Witnesses. We show that if a Prover always sends a batch of proofs containing the same Witnesses, the Verifier will not accept this batch more than 6 times. From the seventh time that a Witness generates a proof, its value is 0, so the Verifier will never accept the batch. The Prover is obliged to renew its Witnesses since the old ones no longer have any value.

Simulation discussion. Following the results of the simulation, we conclude that SureThing will be able to prevent collusions in a real world scenario by taking advantage of the diversity of Witnesses. The decay of a proof value given by the same Witness to the same Prover will eventually deny access to the service. SureThing can also adapt to the number
of users in the place and take advantage of it. It uses the total number of users in the space to calculate the probability of a Prover encountering the same Witness and adjusts the proof decay accordingly.

6 CONCLUSION

In this paper we presented Sure/Thing, a location proof solution for mobile devices, that allows users to prove their location in order to gain access to valuable services provided by other entities. We made Sure/Thing a flexible solution regarding location estimation, by allowing the use of geographical coordinates, Wi-Fi fingerprinting or Bluetooth Beacons. We also proposed different witness models to fulfill different scenarios. We envision that Master witnesses will be valuable to achieve more trusted proofs while Mobile witnesses can still be used to avoid bottlenecks in the system and in situations where there are no Master witnesses. We implemented a last-resort weak model where the Prover can build a proof for himself. This model should only be used in services with less value at stake. The evaluation in this paper, with performance measurements and a simulation scenario, demonstrates that Sure/Thing can be a useful and adaptable framework for using in real world scenarios, where location assurance is increasingly more important in IoT deployments.

Future Work

We think that the next most important step is to increase the level of privacy and security. Since users often share their locations, it is important to protect their privacy. The use of changing pseudonyms is being used in similar systems and can be added to Sure/Thing, to provide stronger privacy protection. The communication between entities is also not protected from eavesdropping. The attacker cannot change the content of a proof, since it is signed by the Witness, but he can listen to its content. Our system will have to adopt cryptographic methods to protect communication and ensure confidentiality.

Sure/Thing can receive new location estimation techniques to be used in location proofs. Sure/Thing would also take advantage of proximity verification methods, to further guarantee that Prover and Witness are really near each other [13]. The renewal of digital certificates is not incorporated in the current prototype of Sure/Thing and it is also a future need. It is important to reward witnesses when they help other users. While this feeling of community already exists in Sure/Thing, we envision that an incentive scheme engine (e.g. gamification) will help developing the community. We proposed a first approach to deal with collusions and evaluated it with simulations. Actual deployments will further validate the effectiveness of the proposed mechanism.

REFERENCES