**Thesis Idea**

**Goal:**

1. ~~Ensure that only legitimate users in target areas are eligible to register.~~
2. ~~Easy for users to register that as many participators as possible can take part in the task.~~
3. ~~Lower the burden of server infrastructure.~~

**Threat model:**

**Devices:**

1. Legitimate users
2. Legitimate users with erroneous devices
3. Malicious users with rooted devices and emulators. No evasion.
4. Malicious users with rooted devices and emulators. Evaded with the help of well-known software.
5. Malicious users with complete customized ROM or emulators.
6. Malicious users with lots of real devices.

**Registration:**

1. Legitimate users
2. Legitimate or malicious users try to register several accounts with one devices.
3. Malicious users try to register several accounts with multiple devices.

**Assumption:**

1. Users should have their own devices.
2. Users’ accounts are combined with IMEI, only one those devices which are combined can a user login.
3. Only those attackers exploit all device detection can have multiple accounts, login them on virtual devices(emulator detection and IMEI spoofing) and contribute corrupted data(root or sensor failure).

**Features can be acquired during registration:**

1. **Phone condition: sensors, root, IMEI number, real device or emulator(also very important),**
2. **Scarce unique personal info: phone number, email addr, personal id.**
3. **Environment condition: IP, network condition, location, devices in the same network.**

**Goal:**

**Stop malicious users or ignorant users from sending false, forged or erroneous data**

**Stop malicious users from registering multiple accounts with one or several identities**

**Here registration doesn’t literally mean the registration process, but users register themselves in the participatory sensing movement**

**Since adversaries can always find a way to exploit system vulnerabilities, we should make sure that registration of several accounts is incredibly challenging without collusion with the same amounts of human beings.**

**one thoughts:**

If we could combine **IMEI number** with **phone number**(e.g. one IMEI number can only be bounded with three or less phone numbers) In such case, if malicious users want to register multiple accounts with one devices, there is a limit. Even if the user could exploit emulator detection, as long as the IMEI number is not spoofed, the cost for registering several accounts is great.

The key of this thought is that we use **two factor authentication**. To be rather extreme, we could maybe ask the user to record voicing a serial of numbers. This clip would be used to generate one time hash( meaning no others could use the clip to analyse the user and this protects the privacy). By comparing hash values, we could know that there is only one user here in the system.

**Aspects of work:**

1. Registration methods. This could apply to all kinds of registration. Referring to Google might help.
2. Authentication method from server side. How to tell human apart from ~~bot~~  (not only bots) .
3. How does adversary evade rooted devices(and the countermeasures to them).

Client side:

Most of the work like detecting sensors or other peripherals are not so difficult. However, have to dig into rooting detection, like mentioned in the thesis. Maybe can detect rooted info from kernel layer with C++.

~~Sever side:~~

~~Spring boot and database implementation. Learn how to implement backend server in one month. General idea is: mobile side sends its testing requests and server sends back the challenge. Between client and server, the message should be secured and cannot be spoofed easily.~~

We want to separate low quality eligible users from strong malicious spammers.

**How could malicious users hide their real identity?**

**Root, IMEI serial number .... (supplement)**

Each of them we should have the countermeasures. Could be from both client side and server side.

**CAPTCHA farm**

Challenges are redirected to developing countries where cheap workers could be found. They solve the CAPTCHA challenge manually and send the response token back to the bot developer.

Depending on the responding time, we could generally determine whether it’s a bot or CAPTCHA or a normal man.

**About CAPTCHA**

As far as I know, it’s not possible or pretty hard to fake sensor data (e.g. temperature, accelerometer, etc.) Therefore if we use sensor to do the CAPTCHA challenge, we just need to guarantee the device is not an emulator.

Previous thesis is doing CAPTCHA challenge locally but it’s not really practical to do it locally considering the storage CAPTCHA may take.

Realize that it’s pretty hard to evaluate the system security level especially botnet detection.

Implementation of the CAPTCHA:

Server generate picture or some random sequence for frontend to generate the captcha test. Picture can be transformed into String format.

CAPTCHA is a simple but not well-developed solution the Sybil attack prevention

Look into some matured Sybil attacks solution and get some inspiration.

Sybil detection in SybilLimit and SybilGuard are based on the fact that Sybil nodes already took some actions(hop) in the network. Then they could whether a node is a Sybil node or not.

The system could require users to establish connections to someone he/she may know to make his or her data becomes effective. However, PS system are not particular distributed system.**Literature Overview**

**Recruitment Framework for Participatory Sensing Data Collections**

Mobility models:

**Location Summary for Personal analytic:**

Consecutive location points within a certain time period are aggregated

into clusters. Can be used to locate user.

(Would this apply to the separation ?)

**Location Prediction and Location based Services:**

These models take a very microscopic view on mobility, concentrating on determining which “cells” a user might travel based on transition patterns from previous cells, time spent in the current cell, and speed/trajectory information

Related papers:

16. Bhattacharya, A., Das, S.: LeZi-update: an information-theoretic approach to track mobile users in PCS networks. In: Proceedings of Mobicom, pp. 1–12. ACM, New York (1999)

17. Soh, W., Kim, H.: Dynamic guard bandwidth scheme for wireless broadband networks. In: Proceedings of Infocom, pp. 572–581. IEEE, Los Alamitos (2001)

**Android Rooting: Race between detection and evasion**

Xposed framework, Java hook to achieve root evasion.

Hook the detecting methods, and execute any code. However only from Java level, very hard to evade from native layer.

Deploy root detection from both Java and native layers

**Android Rooting: Methods, Detection and Evasion**

Mention 7 detection methods(Mostly from Java level). However all of them can be evaded.

Checking common rooting apps: hook package checking methods and hide those sensitive package symbol.

Files or file path detection: hook methods and return fake path detection results.

Build tag detection: usually build tag retrieve from *android.os.build* would be “release”. However this can be modified by Java reflection. Not to mention release tag can also appear in some rooted image.

Check System Property:

....(Other three detection methods)

With hooking method, Hook API would appear at the top of the calling stack of the exception, which indicates that the device must be rooted. (This could be hided if root cloaker tamper with Java runtime

In addition, usually all detection are stateless, which means only return values are required. This makes convenience to root cloakers since they only have to hook the function calls and return fake results. If some shell or pipe which would not trigger root evasion, it would be harder for cloaker to evade. In such condition cloakers have to examine both input and output of the pipe.

(Maybe instead of directly call function Runtime.exec(‘su’), it’s better to first build a shell process using ProcessBuilder.start() to generate a pipe)

**A New CAPTCHA Interface Design for Mobile Devices**

Published back in 2011, therefore a little bit early. Only mention distorted text CAPTCHA and I think there was not sth like ReCAPTCHA back then. Make use of similar animal models to generate a CAPTCHA image. However so far, I think this could also be easily solved by machine learning to tell the difference between animals.

**BeCAPTCHA**

Published in 2015, relatively new paper. Use data set, containing human samples and synthetic ones to train Generative Adversarial Network.

(The result of the research is a little bit difficult to understand)

**Gesture based CAPTCHA**

Distorted text recognition is very similar to Google’s ReCAPTCHA. To make it more sophisticated and user friendly to mobile users, they implement gesture-based CAPTCHA which require human to interact with mobile devices according to the instructions stated in the “distorted sentence”

Comparing to ReCAPTCHA, the success rate is higher, however cannot be proved more strengthened comparing to ReCAPTCHA, not to mention v2 or 3rd generation.

**Trust Management and reputation systems in mobile Participatory Sensing**

**Section3**: adversarial method

**Citation 36 37** talks about how adversarials can manipulate and fool system to gain reputations higher than they merit

**Citation 44** talks about differences between normal applications and a PS one.

**Page 51** three different kinds of PS system

**Citation sentence:** these systems are vulnerable to erroneous contributions

as well as to contributions from malicious participants.

**Might need to read these papers:**

In this section, we present a set of well-known attacks

in networked applications (e.g., [29,54–57]) and discuss how

they apply to participatory sensing applications [25].

**Threat model:**All data-poisoning attacks require sending corrupted data to the system.

-> Malfunctioning software or hardware sensors; Or irrelevant data.

Or sensor or software are exploited

To implement some of the attack, multiple accounts need to be registered with one identity. Or, adversaries could collude and operate a large **collusion attack.** This is basically the same as one owning several legitimate devices (different identities) and then register accounts.

**SybilGuard: Defending Against Sybil Attacks via Social Networks**

Centralized authentication could greatly solve the problem. Collaborating with a government and require secure ID number during registration could possibly solve this. However, this also establish a barrier between the system and legitimate potential users.

Facebook great loss after announcing requiring real identity in the account.

<https://www.theguardian.com/technology/2012/aug/02/facebook-share-price-slumps-20-dollars>

**Hunting Sybils in Participatory Mobile Consensus-Based Networks**

**Threat model:**

Malicious users are going to implement Sybil attack with emulators rather than real devices. Real devices are really costly.

**Users model:**

Honest nodes: honest users with physical devices

Malicious nodes: malicious users with physical devices

Sybil nodes: malicious users with emulated devices

**Two main approaches:**

1. Validate the proximity or triangulate devices location to make sure that they are in the network (This idea is from VANET) (citation 16, 17)
2. Nodes vouch for each other to attest the presence of other nodes and form a graph of evidence, and perform Sybil detection in the graph( citation 7, 19)

**Section 4 Proximity graph construction**

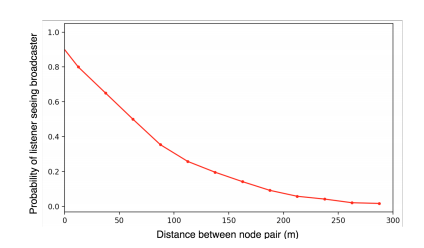
Finally it will build up a matrix:

[N number of nodes][number of rounds]

For the first node, it would be like [‘idle’, ‘broadcast’, ‘idle’, ‘broadcast’,.....,’broadcast’] indicating what would the node do in each round.

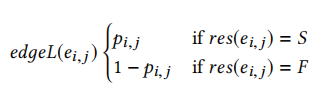
Citation: “Thus after every pair of rounds completes, every node will have had a chance to both see and be seen by every other node from the other group. As will be demonstrated in Section 4.4, these node-to-node connections will be represented

by a proximity graph”

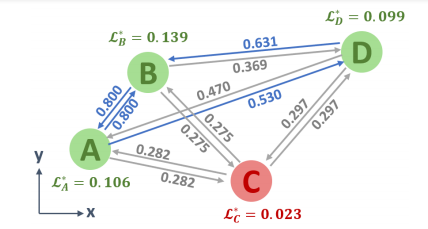


This graph measures the probability of being seen by other nodes and its relation to distance

P\_i,j = P (res(e\_i,j) = S | dist(e\_i,j)) = prob(dist(e\_i,j))



With nearer distance in between, the probability of being seen is high and the further the lower. Therefore we have this graph



Then we could use the following formula to identity

Section 5.3 *Lyapunov’s CLT Approximation* proves that Lyapunov’s CLT can be applied to the model.

The author proves that (l\_vi) the model obeys to normal distribution. Therefore we could then calculate the p-value and those nodes with p-value smaller than 5delta would be eliminated.

2 algorithms:

Elimination one by one

Elimination at one time