Selective Cloaking: Need-to-know for Location-based Apps

Benjamin Henne, Christian Kater,
Matthew Smith, and Michael Brenner

Distributed Computing & Security Group
Leibniz Universität Hannover, Germany

[henne|kater|brenner|smith]@dcsec.uni-hannover.de

Motivation

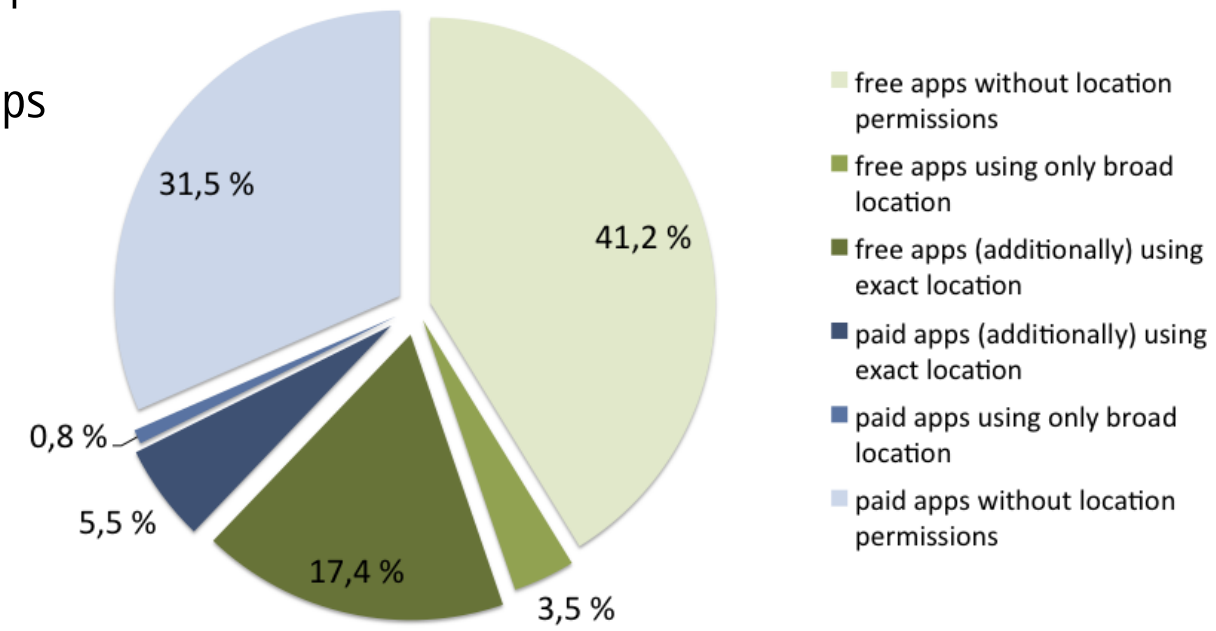
- The still rising adoption of smartphones and tablets entails an increasing use of location-based services ranging from location sharing to the retrieval of location-based information
  - *"18% of US smartphone owners use a geosocial service to* check in *to certain locations or share their locations with friends"*
  - "***74 %** of US smartphone owners use their phone to get real-time location-based information"*

(Pew Internet & American Life Project, May 2012, http://pewinternet.org/Reports/2012/Location-based-services.aspx)

- Very different apps adapt location
  - location sharing, geo-tagging photos or music, local news or weather, a café nearby, cheapest gas station, local radio stations, schedule of next bus stop, local game high scores, ..., fitness, maps, navigation.

Location use of Android Apps

- At end of June 2013, **27.2 % of** 20,681 **Android top apps** found at the Google Play Store on the Web required access to location data
  - 17 % of top paid apps
  - 34 % of top free apps
    - many of those might use location for advertisement



Motivation (2)

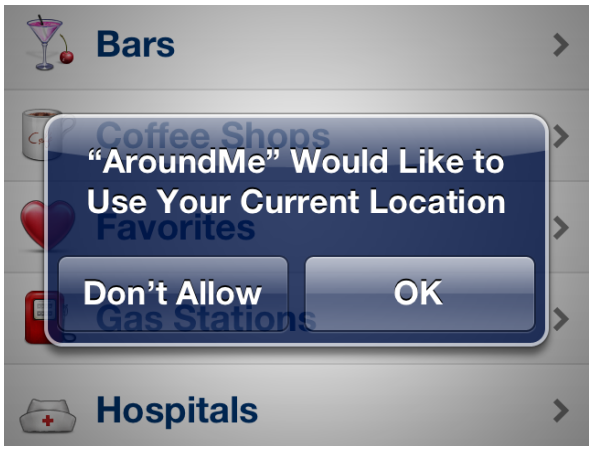
- Current mobile OSs allow users to disable the use of location on a device completely or separately for each app
- However, if users reveal their location to an app, they **always** reveal it with **full precision** even to those that do not need high accuracy
  - e. g. compare: weather forecast service vs. navigation software
    - example: Cyanogenmod 10.1 Lock Clock (cLock)

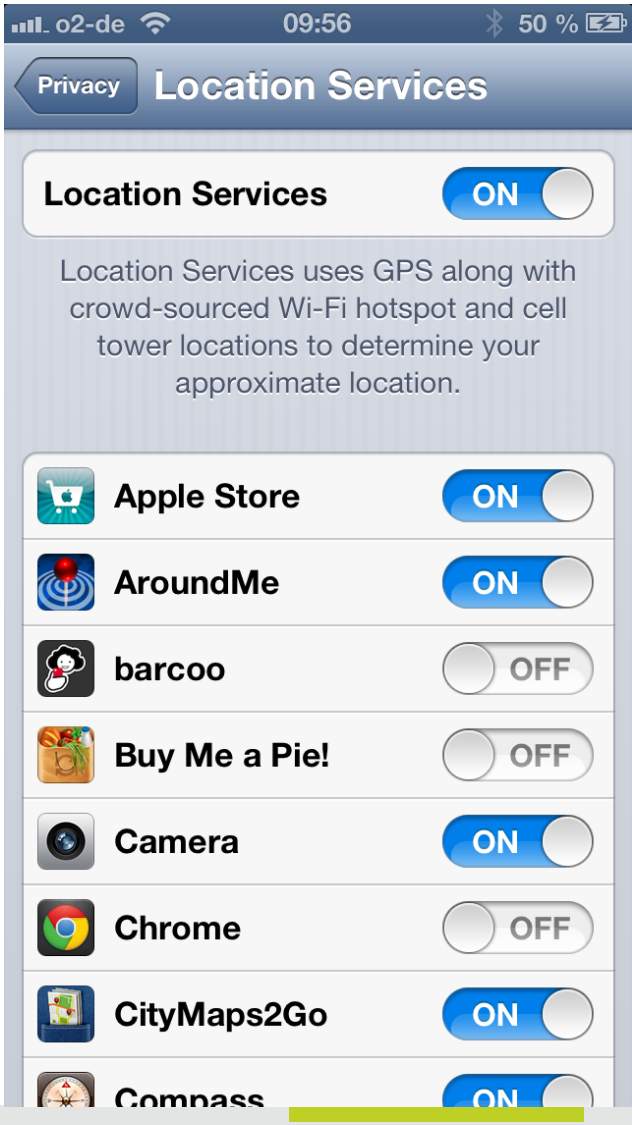
```
HTTP GET
http://query.yahooapis.com/v1/public/yql?
q=select woeid from geo.placefinder
where text="52.379239 9.7229"
and gflags="R“
```

- No mobile OS allows restriction of accuracy of location data by the user

Apple iOS 6

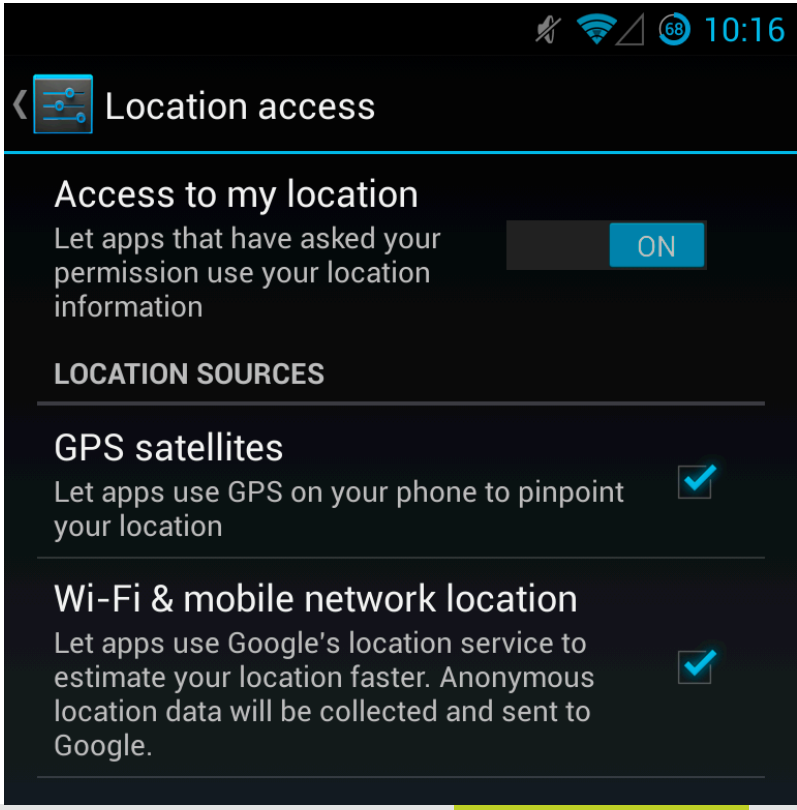
- iOS 6 allows users to enable/disable location use **for all apps at once**
- It includes **selective per-app** configuration
- On first location request of an app, a dialog asks the user about the per-app location privacy setting
  - + including purpose of location request (optional!)





Android

- Android users only can enable/disable location use **for all apps at once**
- Android distinguishes between exact (GPS) and broad (Wi-Fi, cell tower) locations
  - developers define corresponding permissions an app requires to be installed/used
    ⇒ developers select which location exactness their apps receive
  - users only control location sources



Android customizations allows location data restriction

- Custom ROMs and root apps allow restriction of permissions or injection of static spoofed values to prevent privacy
  - *Cyanogenmod 7* – revoke permissions given to an app
    - feature removed[1] again in CM9 when switching to Android 4
    - CM 10.1 introduced *Privacy Guard*, which allows spoofing "no location"
  - *PDroid 2.0* – block access to private data; spoof fixed location
  - *AppFence* – substitute private data by shadow data like fixed location
  - *MyShield* – basic reducing of location exactness:
    untrusted apps get no location data;
    trusted apps get location rounded to one decimal place
    - unpredictable effect: dislocation between 0 m and ~7.85 km

[1] http://forum.cyanogenmod.com/topic/44589-combining-cyanogen-with-pdroid/#entry301937

Location Privacy in Research summed up

- Mobile privacy research proposed different privacy solutions, but those mostly focus on tracking and protecting access to private data in general
  - *TaintDroid*, *AppFence*, MyShield, DroidScope, …
- Privacy research proposed many location cloaking mechanism in the past like shifting coordinates, k-anonymity, local mechanism, service-based, p2p, …
  - Mostly theoretical works
  - Missing evaluations of real-world practicability and usability
  - Which are applicable to the smartphone use case at all?
- If users want use location on their smartphones and tables today,
  - there is no possibility for obfuscation on mobile devices
    - ! privacy must not be implemented on server-side only

Our Goal: Controlled Usage and Accuracy of Location

- There are many apps that request location, but might not need it from the users' perspective – disabling access is ok
  - apps with banner ads, features like geo-tagging music in Shazam

! However, there are many location-based apps users want to use, but they still have privacy concerns!

⇒ Since many apps do not require exact locations, we can **improve privacy** by only **disclosing location only** in such **detail as needed** by an app **to function**

Android Location Privacy Framework (LPF)

- Allows Android users location obfuscation for all apps that request location data from the Android system

  - Extended Cyanogenmod 9.1 with Location Obfuscation
    - tested with AOSP 4.0.4 and Cyanogenmod 10.1 also

  Roadmap

  ✓ Framework for easy integration of algorithms on devices as 1st step

  ○ Usable privacy solution will be the 2nd step – future work

Android Location Privacy Framework (LPF) – Features

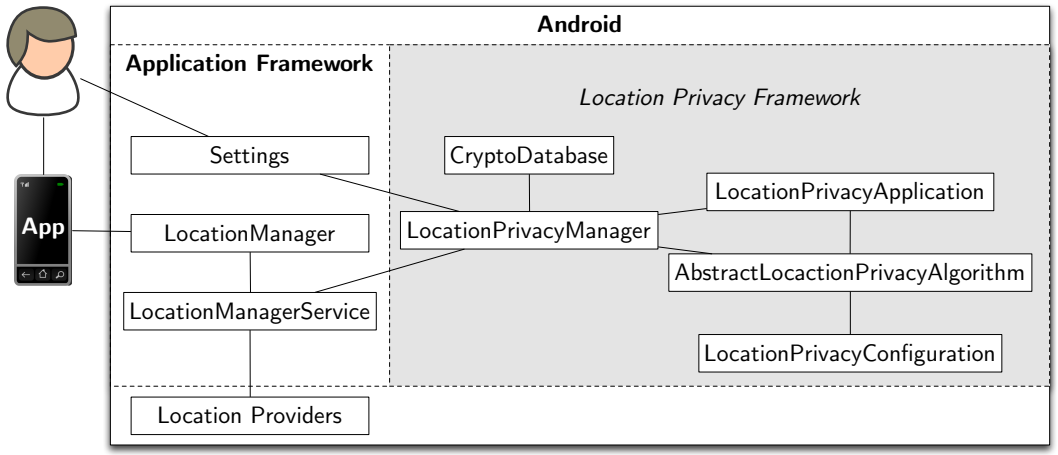
- Allows Android users location obfuscation for all apps that request location data from the Android system

  From the user perspective

  - Basically, obfuscation is centrally enabled/disabled for all apps at once
    - Any app
      - is discovered by LPF when it requests location the first time
      - uses the configured default obfuscation by default – privacy as opt-out
  - Users can configure app-specific obfuscation
    - Enable/disable obfuscation
    - Select algorithm
    - Set parameter values

Integration of LPF into Android

- If an app is granted location permissions, it can request location data
- It queries location data via an *LocationManager instance*
- Locations are acquired in different ways from *LocationManagerService*
- *The service gets location data from different location sources (providers)*
- *Each locations passes one of two methods in the service*
  - *here location obfuscation is applied before an app receives the data*



Framework integration in *LocationManagerService* at `getLastKnownLocation()` and `callLocationChangedLocked()`

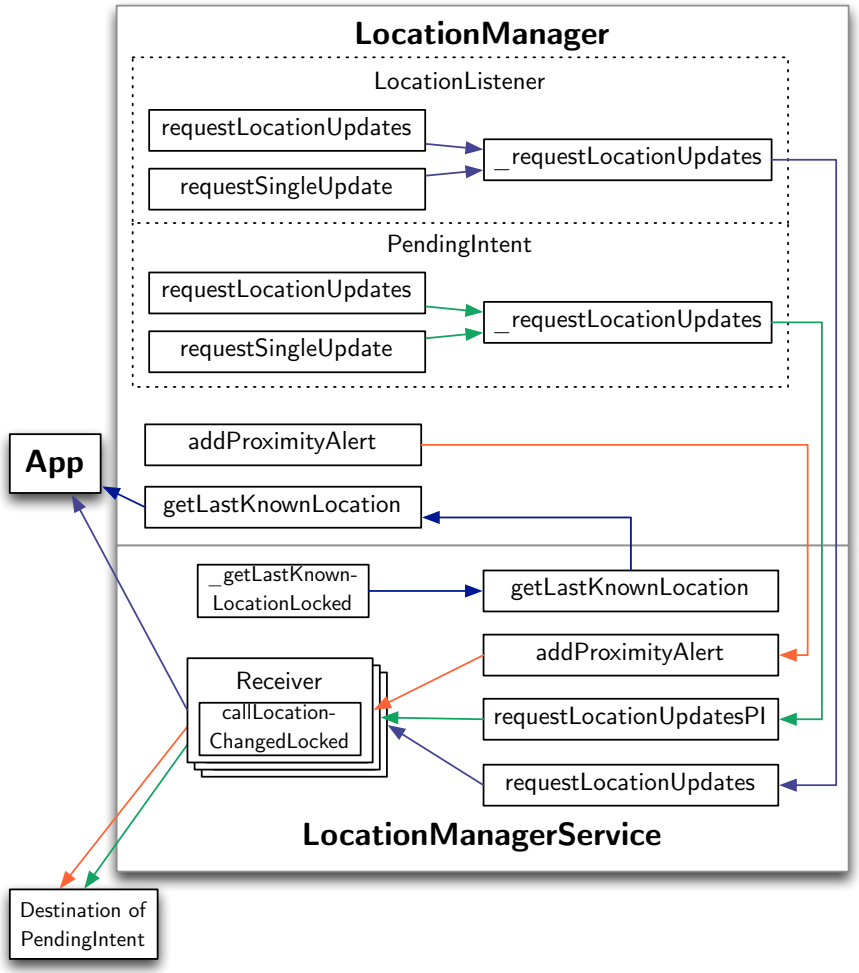
Integration of LPF into Android – deep insights

- Locations can be acquired using
  - calling *getLastKnownLocation()*
  - using *LocationListeners*
  - via a *PendingIntent*
  - within *ProximityAlerts*

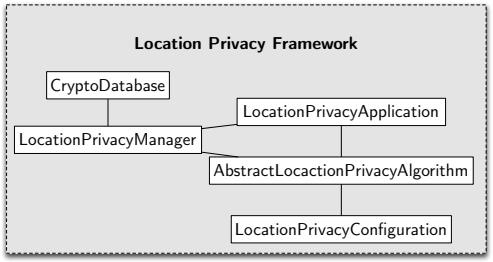
  via a *LocationManager*

- Each location passes
  - *getLastKnownLocation()* or
  - *callLocationChangedLocked()*

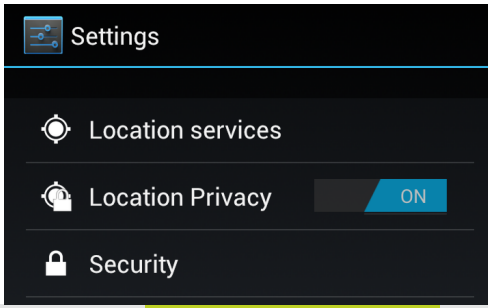
  in the *LocationManagerService*



LPF Components

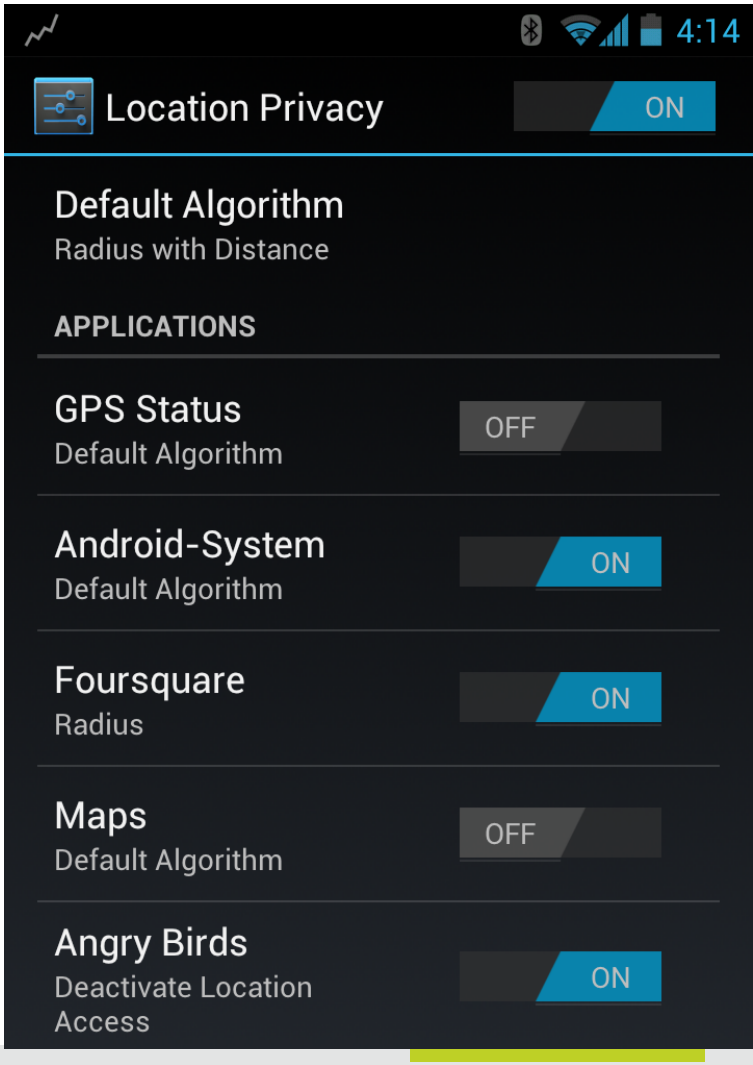


- Model
  - *AbstractLocationPrivacyAlgorithm* – template for algorithm implementation
    - *obfuscate(Location)* implements obfuscation
    - *getDefaultConfiguration()* defines parameters and default values
- Control
  - *CryptoDatabase* – AES-encrypted SQLite-backend for storing configuration
    - Password is generated on first usage and store in OS/root context
  - *LocationPrivacyManager* – all actions for management and obfuscation
    - *obfuscateLocation()* obfuscates a location as configured for the requesting app
      - Apps are identified by *uid* and *package name*
- Settings
  - Android Settings app has been extended
    - integrated with other location/security options



LPF – App-specific configuration

- App configuration is build from detected location apps and algorithm models
  - UI defined by model's parameter types
    - Text fields, number input
    - Check boxes for Boolean
    - Drop-down selections for Lists
    - Map Activity for picking coordinates including geo-coding and Web search
- Localized parameter descriptions via Android's strings.xml



Obfuscation Algorithms

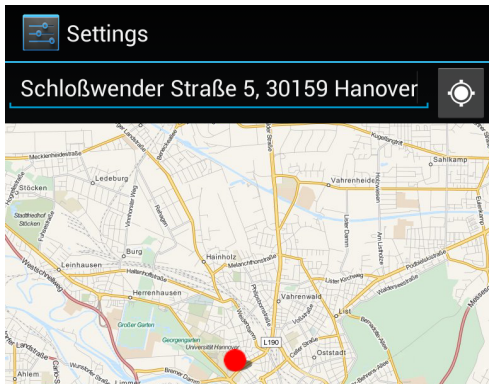
- On-Device Obfuscation
  - Work without Internet connection
    - Do not disclose location to any external service
    - Mostly faster – no network delays
    - Mostly are of simpler nature
  - Might be satisfactory for many use cases
- Service-based Obfuscation
  - Need Internet access – most apps using location do either
  - Integrate external service or external data
  - Allow more complex obfuscation
    - e. g. considering other users' locations, or data like spatial data
  - Might share location with such external services

On-Device Obfuscation

- Deactivation
  - An app receives no location data
  - Like per-app disabling in iOS 6

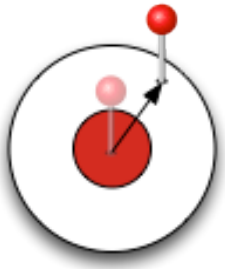
```
return null;
```

- Fixed coordinates
  - App receives a fixed value pre-selected by the user
    - pin on map
    - search address



- Random shift w/max
  - Shift in random direction
  - Distance $0 \leq d \leq max$ meters

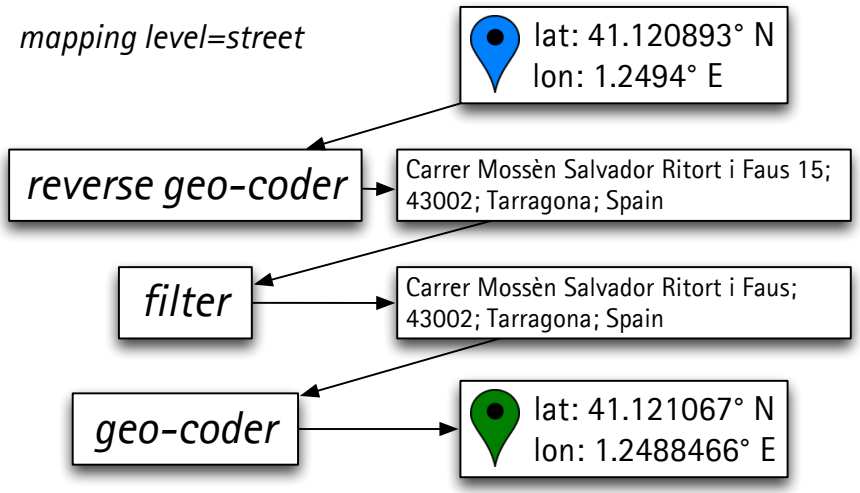
- Random shift w/min+max
  - Shift in random direction
  - Distance $min \leq d \leq max$ meters



- contraint: calculate new location, iff real location changed x meters
  - ➔ *for privacy and usability reasons*

Service-based Obfuscation

- Web service - Query external web service via HTTPS w/basic auth to obfuscate location before it is given to an app
- Geo-data-based Mapping
  - Map Location to center of bounding-box of a geographic datum
    - Level: nearest street, postal code region, city boundary, country



Basic Performance Evaluation

Evaluation on Samsung Galaxy Nexus (ARM Cortex-A9, 1GB RAM)

- compared OS with/without extension: 10,000 consecutively calls to *getLastKnownLocation()* with random shifting took between 4.2s and 4.9s
  - ⇒ **No measurable slow down by the LPF framework itself**
- Algorithms compared
  - On-device algorithms equally fast *max* caused by DB access on first obfuscation after reconfigure
  - Web service slower due to network delay for single Web request to service in our local network
  - Geo-coding slowest due to two API calls to Google GeoCoder

obfuscation algorithm	*min* [s]	*avg* [s]	*max* [s]
Fixed location	0.0008	0.0012	0.0019
Random (max)	0.001	0.0013	0.0005
Random (min/max)	0.001	0.0015	0.0005
Web service	0.2	0.23	0.4
Geo-coding	0.2	0.46	1

⇒ **Obfuscation time is tolerable**
– since location requests mostly done in background

Limitations of System-wide Obfuscation

- Only locations acquired via Android's base system are obfuscated
- Some apps (may) implement own positioning methods

Google Maps does

  - uses GPS location from Android system
  - uses network-based location from Android system
  - BUT additionally uses own network-based positioning implementation

  ⇒ Iff GPS is on and got a fix, it uses system (obfuscated) location

  ⇒ Else, location acquired from independent positioning is used

The worse: *Google Play Services* behave likewise

  - They unify API usage of multiple Android versions for app developers, BUT exclude privacy enhancements as they are closed source software by Google

Usability Hurdles of Location Obfuscation

- Many obfuscation techniques might be hard to understand for users
- Users might not be able to set up appropriate parameters
  - It is hard to realize what k=20 means in daily life k-anonymity
  - It is even hard to realize how big a radius r=500m is for many people
  - Also, it is hard to determine what exactness each app needs to function
- Users might not be willing to think about technical details at all
  - They might not be interested in different algorithms
  - Interested in obfuscation results
    – but independently of any algorithm concept

Research Challenges for Usable Location Obfuscation

- Select **comprehensible algorithm(s)** that match privacy needs
- Find **concepts like metaphor** to make parameters more easy to grasp
  - like geo-coding algorithm
    – uses common concepts as street, district, city, …
- Identify **generally acceptable parameter values**
  - as defaults
  - as recommendation

Conclusion

- We built system-wide location obfuscation for Android
  - We implemented need-to-know principle (client-side obfuscation), while services do "we should store less" (server-side obfuscation)
  - Compared to many other works that focused on recognition and restriction of data disclosure in general
- The Location Privacy Framework can be easily used to evaluate existing or new obfuscation algorithms in the real mobile ecosystem
- Users are enabled to use location-based apps with different location accuracy depending apps' use cases and real needs
  - Current framework implementation is technical
    - All may use it with appropriate defaults
    - Only some may comprehend details and change defaults

Future Work

- Build and evaluate a Usable Location Obfuscation solution
  - We just started with some focus group discussions last week

<details were stripped in slides online version>

Try the *Location Privacy Framework* yourself

- Source code on GitHub
  http://bhenne.github.io/android-location-privacy/
- ROMs – URLs were stripped in slides online version – please ask for it via email
  - Cyanogenmod 10.1 (LG Nexus 4)
  - Cyanogenmod 9.1 (Samsung Galaxy Nexus)
  - AOSP 4.0.4 (Samsung Galaxy Nexus)
- Screenshot series on YouTube – user DCSecUniHannover
  http://youtu.be/80AzmH60epM