Université de Lille
Master Informatique, spécialité e-services

Improving Android: location privacy protection and energy consumption

Author:
Rémy Raes

Supervisors:
Romain Rouvoy
Lakhdar Meftah
Pierre Bourhis

University supervisor:
Xavier Le Pallec

September 2019, 4th

Abstract

Android, main operating system of smartphones sold throughout the world, allows its users many actions: phone calls, messages sending, web browsing, applications downloads...

The latter are not always respectful of their users privacy, and can learn more information than they ask for thanks to locations; in this context, we present an application allowing users to know who tracks them, information known about them, and finally protect their privacy.

In an rich ecosystem with millions of applications whose two-thirds of the source code consists in calls to external libraries, it is important to monitor their functioning, to be able to choose the most efficient: we propose to compare libraries among the most used by developers.

Contents

Part 1

Protecting users privacy

1.1 Introduction

In 2019, smartphones do well: we count 3 billion users throughout the world (against 1.86 billion users in 2015 [19]). Thanks to this platform, it is possible to use applications with several types of use: social networks, GPS, games, messages... Unfortunately, using these applications is not free; even if a part of these can be installed on a smartphone without spending any money, their publishers gain money in another way: they gather personal data from their users.
In a more or less open way, these editors (Google and Facebook, for example) gather data on their applications users which are more or less personal: chat history, photos, and even geographical locations.
This is a major problem. Indeed, if you regroup locations gathered by applications on a given time interval, it is possible to determine which places a user spends time on; and by cross-checking these places with a map, it is easy to speculate sensitive information such as commercial habits, place of living, diseases, political ideas or religious involvement.

Thus, we propose a model for a polyvalent Android application.
This application can:

- **analyse application accesses to locations**, e.g. show to the user that Facebook accessed his location 18 times in the present day;
- **evaluate the risks for the privacy**, reproduce known attacks on users data, and display what information can be deduced from them;
- **activate protection mecanisms** as a consequence: our application, acting as a privacy control center, can furnish "safe" locations to applications, in order to guarantee users privacy protection.

1.2 Related work

To make users aware of their data' cost, we reproduce state-of-the-art attacks on users' positions, to show them what private information can be inferred from their locations data.
Once users are aware they have to protect their data, we present them with mechanisms that allow them to set up and benefit from a better protection.

1.2.1 Geolocalisation attacks

Even if Android was released in 2007, researchers have already been working on location privacy issues, proposing papers which, although aiming at GPS technologies, can be applied to Android as well.
Since the 90s, we know how to extract *points of interest* (POIs) from a locations dataset. POIs are physical places where a user is located for a certain duration; they may reveal sensitive information such as place of work, of living, politic and religious beliefs.
There are different ways to extract POIs from data.

K-means is a well-known clustering algorithm, which initially randomly assigns all points to a given number of clusters, and then iterates through all points, assigning them to the closest cluster, until the clusters do not evolve anymore. This approach has been implemented on locations by Ashbrook and Starner[2], but unfortunately does not suit what we want to do, since it requires a number of clusters, which we don't know (*i.e.* we don't know how many places have been visited by a user) and that all location points are included while building clusters, even the ones between real locations (*i.e.* path points), which introduces some noise into extracted POIs locations; for example, the positions on the route of a commuter going to work do not infer information about his home or work place, and thus should be ignored; otherwise, the clusters centres would tend to move towards these "noisy" locations.
Another approach is **density-based clustering**. To define density, this algorithm uses two parameters: the radius of a circle, and the minimum number of locations within this circle. Implemented by Ester *et al.*[6], this algorithm has many advantages over k-means: it forms clusters that are not necessarily round-shaped, it tends to ignore unusual locations for the clustering, and it is deterministic, meaning that it will always produce the same clusters with the same input (which is not the case with k-means, given that the result is strongly bound to the initial random assignment of points to clusters).
This algorithm produces acceptable POIs, but can contain points that are not interesting for us (frequent stops at red lights on the road, for example). To fix that problem, Zhou *et al.* propose DJ-Cluster[21], a density and join-based clustering algorithm that computes neighbourhood of each point, thus removing strong dependency to input settings, and using temporal-filtering techniques to eliminate uninteresting locations.

If users want to protect themselves by sending obfuscated locations, attackers can still determine sensible information by using different techniques and previous knowledge.
Maps can be used to reduce the level of obfuscation[12], by removing all locations that seem impossible (*e.g.* when crossing a river bridge, removing locations situated in the water); in a maximum movement boundary attack[8], the attacker computes a zone where a user can be, based on the time interval between two location updates and maximum velocity of the user.

All these attacks can be led one by one, but can also be combined to better undermine users' privacy.

1.2.2 Users protection

Nowadays, all smartphones are equipped with GPS chips, allowing the use of *Location-based services* (LBS) such as GPS navigation and location-based games, thus providing some benefits to users, but raising privacy issues by allowing companies to use location-data mining algorithms, as described above.
To protect users from such threats, *location privacy protection mechanisms* (LPPMs) are employed. LPPMs generally take a set of locations as input, and produces a set of obfuscated ones. They can be used in two different ways: as **online** mechanisms, when each location is modified before being sent to external services, or as **offline** mechanisms, where the entire set of locations is obfuscated at once.

Andrés *et al.* propose Geo-Indistinguishability[1], a strategy that allows a user to hide his real location into a pre-determined-radius-wide area, by providing the algorithm a radius and the level of discrepancy that he/she can tolerate between the likelihood of the various points in the area (which are candidate locations from the LBS point of view), thus introducing spacial noise; to add temporal noise to the data, and to prevent the extraction of POIs while maintaining a good spacial accuracy, Primault *et al.* developed Promesse[16], which both distorts locations timestamps and adds new locations to the dataset in order to keep a constant distance between two points of the set; to ease user manipulation of these protection mechanisms, Cerf *et al.* propose PULP[3], a framework which aims at selecting a suitable LPPM according to users' objectives and automatically configuring it, using non-linear models; they also propose a control-theoretic approach to study and assure a better control over existing LPPMs[4].
As proposed by Samarati *et al.*[18], **k-anonymity** allows to hide a user in a group of *k-1* users, *i.e.* in a geographical context, making a user indistinguishable from a number of users in a given zone. An implementation of it is CliqueCloak[7], which uses a trusted external server to create virtual cloaking zones; however, if the trusted server is compromised, locations of all its users could be revealed: PRIVÉ[9] addresses this issue by organising users into a peer-to-peer network, thus avoiding the bottleneck induced by centralised techniques.
All above LPPMs protect the privacy of the users, however they do not block re-identification attacks such as AP-Attack[13], an attack that do not focus on a subset of points, but rather aggregates all the points enclosed in a user mobility trace into a heatmap structure. These attacks are being able to associate from 63% to 89% of obfuscated POIs with real ones[15]. To counter that, new LPPMs are proposed, such as HMC[14], which builds a heatmap from the user mobility trace, obfuscates it, and then reconstructs a mobility trace from the altered heatmap.

1.3 Privacy Control Center

We propose an Android application that enables users to see which data LBS have about them, which dangers they front while sending locations to LBS, and enables them to set up protections to prevent such dangers from harming them.

Regarding the application itself, it has been made using fragments, reusable pieces of application that allow us to produce more human-readable source code and a more organised architecture for the application; some of its components have been unit-tested to ensure good functioning in any situation (*e.g.* with any input data).

1.3.1 Data gathering

LBS gather users' locations on several occasions and at various times from their smartphones directly; we present here how we access this locations gathering process, and how users can visualise it.

1.3.1.1 Android location operating

To get users' locations on Android, developers have access to several location providers:

- **GPS** using the GPS antenna of the smartphone;
- **Network** using other captors: Wi-Fi, celular networks, BlueTooth;
- **Passive** not using any hardware, but rather transmitting a location each time one of the previous providers receives one (*e.g.* when Google Maps requests a location, the provider also receives the location object)

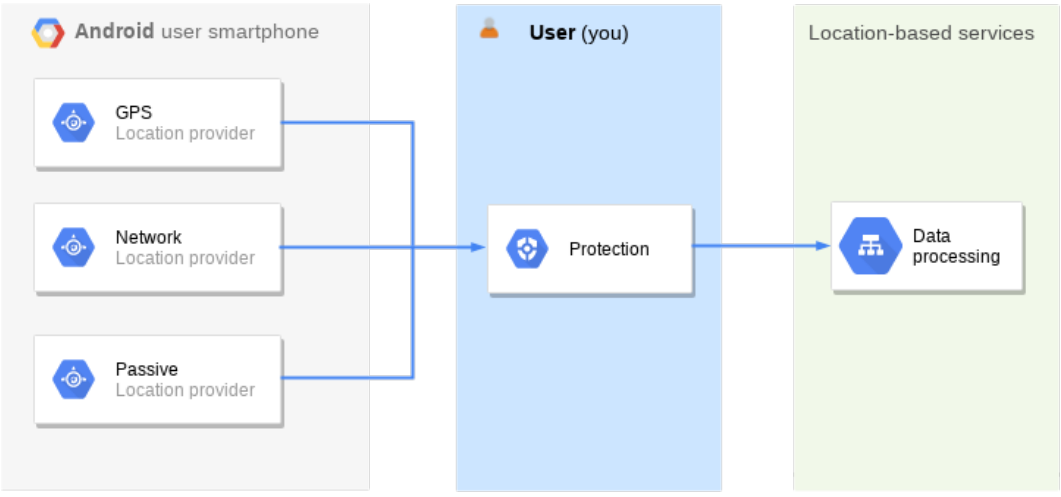


Figure 1.1: Global diagram of location providers working

To develop our application, we use the last one, which fits well what we want to do.

Indeed, we don't want to ask the smartphone for locations in an active way, but rather want to listen and gather locations that have been asked for by others applications on the smartphone.

*N.B.: Android gives access to another location provider, the **FusedLocationProvider**; as its name hints, it regroups previously mentioned location providers, and allows developers not to implement a mixed-policy of locations gathering between GPS and network accesses.*

1.3.1.2 Data visualisation

We implemented two different ways to visualise information (*e.g.* locations gathered to feed LBS).
The first one is a *heatmap* (see figure 1.2) which represents locations on a map, using a color scale (from green to red) to represent the user presences intensity in a given zone.
The second one is a list of all gathered locations, that can be handled by the user following time criteria; the user can also display one particular location as a marker on the map.
All data gathered by the application are stored locally, on the smartphone : the user can dispose of it as he/she pleases, and can delete specific locations if he/she wishes to do so; uninstalling the application triggers all databases removal, and thus deletion of all gathered data. (to import fake locations and ease the application development, we implemented import/export functions, allowing us to gather data on the ground with personal smartphones and import it into the team's smartphones)

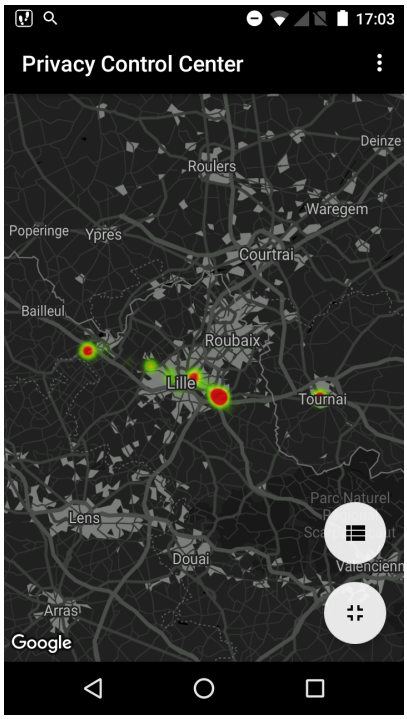
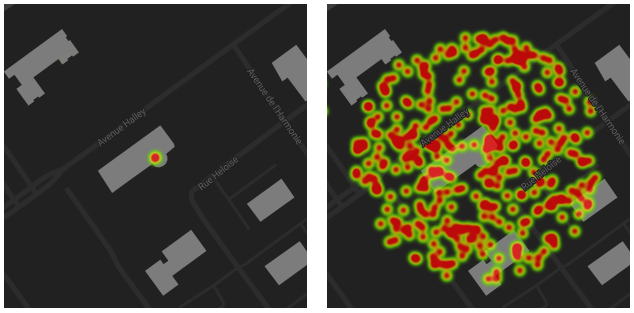


Figure 1.2: Screenshot of the visualisation screen

1.3.2 User protection

Our application uses two types of filters, one being temporal, the other geographical; due to our application acting as a live firewall filtering locations as they arrive, we only use online LPPMs as described in the state of the art.
The first one allows the user to create time slices (with a starting time and an ending time), within which locations transmission to applications can be authorised or not.
In a similar way, the second filter allows the user to draw geographical zones on a map, to name them, and to determine if locations capted in these zones can be transmitted to location-based services or not.
After the gathered location has passed these filters, and before sending it to the application requiring it, we add a random noise to it, similarly to Geo-I[1]; a dedicated setting in the application allows the user to vary the radius of the zone (in meters).



(a) Before adding noise (b) After adding noise

Figure 1.3: Example of noise with a radius of 100 metres

Now that we have our location object altered to protect the user privacy, if it is not caught by any of the previously-mentioned mechanisms, we can send it to the application that asked for it.
To do so, we put our application as a location provider for other applications on the smartphone, thanks to a developer option[1].

1.3.3 Simulating attacks

To make the users aware that revealing too much information about themselves can be harmful, we execute well-known attacks on their gathered data, hence showing the users what application editors can learn about them from their locations.
In our application, an interface allows users to not only launch these algorithms against their data, but also to activate previously-mentioned LPPMs to see their protection effects live, as demonstrated in figure 1.3.

1.4 Evaluating the application

Aiming at both finding threats to users privacy and enabling them to fix these threats, our application allows the users to produce as many times as wanted a group of actions that sets users as main actors in their data protection, this privacy protection cycle is represented in figure 1.4.
First, we perform previously-described attacks on user locations history; then, we display the information we learnt and can extrapolate about him/her; finally, the user adapts the LPPMs, and rerun this cycle.

[1] `https://developer.android.com/studio/debug/dev-options#debugging`

Figure 1.4: Privacy protection improvement cycle

1.5 Discussion

1.5.1 Bias

On the Android ecosystem, location providers only transmit Location[2] objects, which contain latitude, longitude, gathering time stamp, accuracy on different scales, but not the identity of the application that asked for it; this is understandable regarding the system security, but prevents us from showing easily to users who is spying on them.

Moreover, the protection supplied by our application does not apply to all locations by default: actually, when it is installed on a smartphone for the first time, no protection zones or time slices are present, meaning that locations aren't matched by any filter; should they be transmitted when this happens?
If the users can define zones where locations transmission is forbidden, they can also define zones where it is allowed; then, what happens to locations gathered outside of any zone ? This strongly questions the utility of these ban zones.

1.5.2 Future work

1.5.2.1 Data utility

If our application allows to protect the exact location of its users, it raises a new issue: data utility.

In practice, transforming a geographical datum also means removing the information initially supported by it. For example, if the user sets the noise filter value to 100 meters and wants to use a GPS application on his smartphone, his virtual location is situated in a 200-meter-diameter

[2] `https://developer.android.com/reference/android/location/Location`

wide circle around his real location, which is enough for the GPS app to locate him on another street, even in another district, thus preventing its normal functioning.
Unfortunately, this issue cannot be solved with an algorithm or a piece of software, it's a communication issue that has to be solved; we have to convince users to go against their instinct of activating all protections to maximal level, so that their privacy protection still allows them to use their favourite applications.

1.5.2.2 Attacks

In this paper, we only are using location mining techniques, effectively allowing us to find POIs, but ignoring paths between those, while they can be used to find out even more information, including, for example, which transport users employ to go to work.

1.5.2.3 Application evaluation

When evaluating protection settings selected by the user, we rely on a set of actions (described in figure 1.4) that includes the user, thus not implying that the level of protection necessarily rise with each iteration of the cycle: use of re-identification attacks such as [13] is required to better evaluate the application protection efficiency.

1.5.2.4 Protection detection

Finally, regarding the noise added to locations, its randomness can betray our application to LBS consuming these locations. Since randomness of the noise is computed at each transmitted location, even if this conceals real locations, it can lead to production of incorrect data if the user does not move.
Moreover, this approach effectively cloaks users while used online, but it is evident that LBS store and have access to whole users mobility data sets, thus being able to see that we are trying to trick them.

Part 2

Studying the impact of third-party libraries on applications

2.1 Introduction

As seen in the previous chapter, smartphones market is considerably growing. Users count increase is linked to the fact that populations in disadvantaged areas are coming online for the first time, and gain access to smartphones and new technologies, thus creating new emerging markets.
However, part of these new users face constraints that are not or not enough addressed by developed markets: limited bandwidth, connection cost, screens size, amongst others.
To address these issues, Google implemented a series of good practices consisting in optimising applications available on the Play Store: this policy is called *build for the next billion*[10]. When developing applications, these practices include using small amount of RAM, being responsive to small screen sizes, reducing both energy consumption and size of the APK (*Android Package Kit*, which is the final file downloaded by users on the Play Store), for it to be downloaded faster.
As said previously, the smaller the application size, the bigger the number of users. An important part (up to two-thirds[20]) of the APK file is made of third-party libraries included by developers. There are thousands of libraries, for any types of usages, from advertising providers to audience analysers, by way of testing solutions.
Because there are several (actually, lots of) libraries for the exact same uses, and because it is sometimes difficult to compare them, we propose to compare few of them based on criteria following this policy of creating application for the many. Tested libraries cover three categories:

- advertising;
- analytics;
- crash reporting.

By studying and comparing consumption of what makes up the majority of modern applications, we wish to be able to provide new criteria for developers to choose the libraries they use in their projects.

2.2 Related work

Our contribution completes a previous work, made by *Rubén Saborido, Foutse Khomh and Yann-Gaël Guéhéneuc*[17]; in this work, they study impact of third-party libraries on Android applications, their approach consists in running scenarios on minimal apps and using different scripts, estimating different metrics: CPU, memory, and network usages, number of permissions, energy consumption, and app size.
However, they found some inconsistencies in their measurements. They used to have access to a piece of hardware they designed to measure energy consumption, but that is not possible anymore, therefore showing our help could be valuable in this study, our team fortunately having such hardware in stock.

2.3 Settings

Here are the libraries we experiment on; since we use APK files generated by our colleagues, we are testing the same library versions. Also, to better stick to criteria of the experimentation led by our colleagues, we use a LG G4 smartphone, which we downgraded to Android 5.1.

Name	Version	Provider	Category
admob	10.2.4	Google	Advertising
applovin	7.2.0	Applovin	Advertising
mopub	4.15.0	Twitter	Advertising
firebase	11.0.2	Google	Analytics
flurry	7.0.0	Yahoo	Analytics
google	10.2.4	Google	Analytics
acra	4.9.2	ACRA	Crash reporting
crashlytics	2.6	Crashlytics	Crash reporting
newrelic	5.14.0	Newrelic	Crash reporting

Figure 2.1: Tested third-party libraries

To test previously mentioned third-party libraries, we use minimal applications which contain nothing but components strictly needed for the tested library to work.
We then run pre-established scenarios (see appendix for example) for each of the libraries, which launches the corresponding activity, and simulates user interactions with `touch`, `swipe` and `sleep` commands; these scenarios had to be updated with our smartphone's components coordinates, due to it having a different screen resolution (*e.g.* the back button wasn't properly specified).

2.4 Measurements

2.4.1 Tools

We experimented with three different measurement methods:

- Android Studio
- Battery Historian
- Monsoon probe

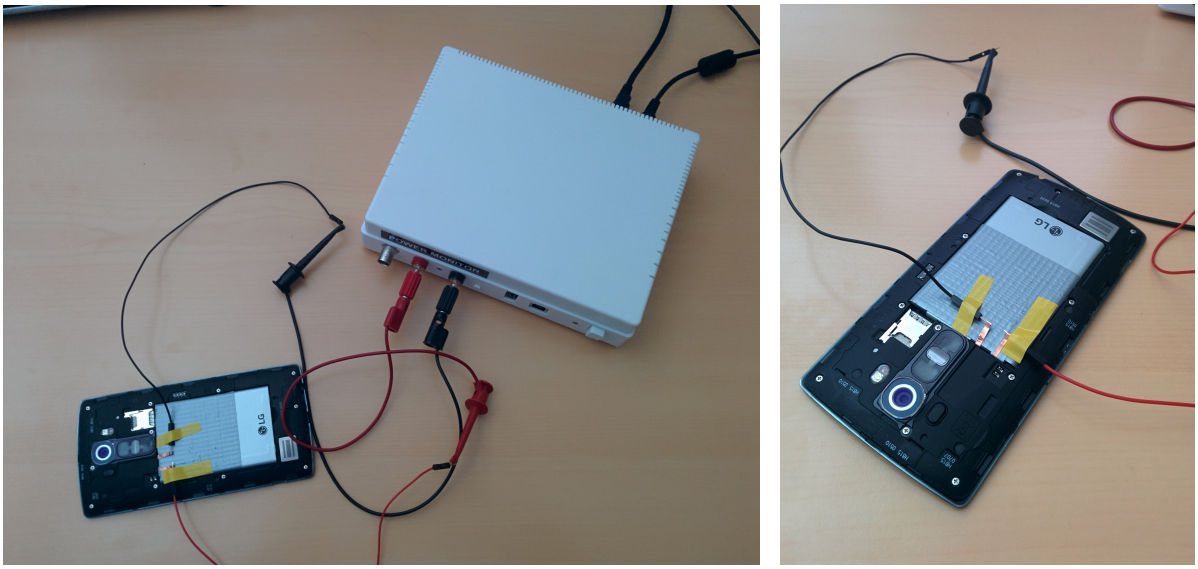
Android Studio is the privileged IDE for many Android developers, as it is free-to-use and integrates many useful tools to develop and debug applications. One of them is Android Profiler, a built-in tool that enables developers to view CPU consumption, network and battery use in real-time.
Unfortunately, the granularity of the battery profiler is too important to provide us with meaningful results, and cannot be used as such.

Battery Historian is also a Google-developed tool that allows developers to inspect battery-related events on an Android smartphone, while it is not connected to the computer (the analysis is done by exporting a report from the device, and then using it as input with Battery Historian). However, the last commit on this project is 2 years old, it requires Android 7 smartphones, and provides us with a thinner granularity than Android Studio, but not enough to notice energy consumption over a few minutes of usage.

The solution we use is a **LVPM** (standing for *low voltage power monitor*), it is a probe that enables measurements on either physical or USB channels.
Although the LVPM is not supported anymore by its manufacturer, its deprecated interface still works (on Windows); plus, some components of the Python library (built for the high-voltage version of the probe) are working on the LVPM.

In order to measure consumption, we have to bypass the device's battery to avoid inconsistencies linked to it. We tried to connect the power monitor directly to the smartphone's pins to supply power to it, but it seems that if all battery pins (positive, negative, BSI, BTEMP[3]) are not connected, the smartphone does not boot up: we have to make a little electrical assembly on our device's battery.

[3]BSI stands for *battery status indicator*, BTEMP for *battery temperature*



(a) Power monitor (b) Bypassing battery pins

Figure 2.2: Electrical montage

2.4.2 Approach

The previous montage only provides measured voltage input to the smartphone. To run our scenarios on it, we need a USB connection, which would be another source of energy, and thus would skew results.
To solve this problem, we have to root[4] the phone, and then install an application that effectively deactivates USB charging.

To eliminate inconsistencies in measurements linked to cables resistance, we roll each scenario 30 times, and compute the average consumption.

2.5 Discussion

2.5.1 Bias

We do not have the exact same platform our colleagues worked with, since they used a Google Nexus 4 model while we use a LG G4, which could lead to inconsistencies since both battery models are slightly different; at first we wanted to run scenarios using a Google Nexus 5, but unfortunately the battery connections are poorly located (the battery does not have pins and is unremovable).
At the time of the study, we only had access to minimal application's APKs, which ensured testing same libraries versions, but prevented us from testing two advertising libraries, these requiring a certain device ID to load ads.

[4]Rooting is the process of allowing users of smartphones [...] running the Android mobile operating system to attain privileged control (known as root access). - `https://en.wikipedia.org/wiki/Rooting_(Android)`

2.5.2 Future work

When finishing this thesis, we just had rooted the smartphone, meaning that results will not appear in this work, but should be included in the final viva.

Appendix

Haxe

While starting to work on the subject, we aimed at using a multi-platform framework, to bring the application to several modern platforms (Android, iOS, Chrome).
We headed towards Haxe, a technology enabling multi-platform development. At first sight, we perceived that the language support was limited: its developers having chosen a "gaming" approach for it, application interfaces libraries are at best almost working, at worst not at all, being several years old in all cases.
After successfully displaying a matrix (see figure A.1) with a graphics library, we looked for components enabling us to display a map (critical fonction of our application); the only official library being deprecated, and all custom users implementations not working (CPU-intensive, tiles not displaying...), we decided not to use Haxe for our project.

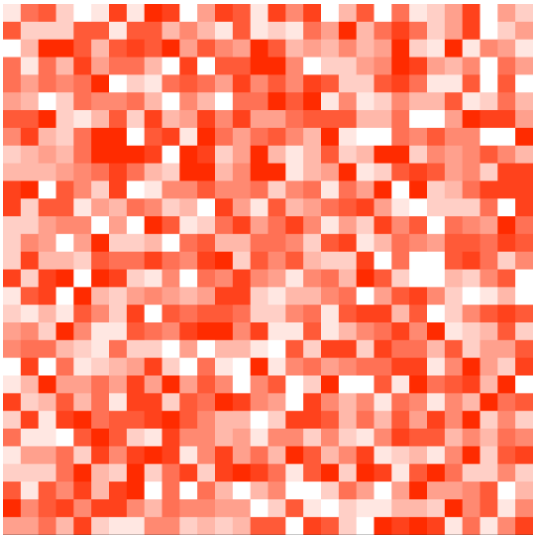
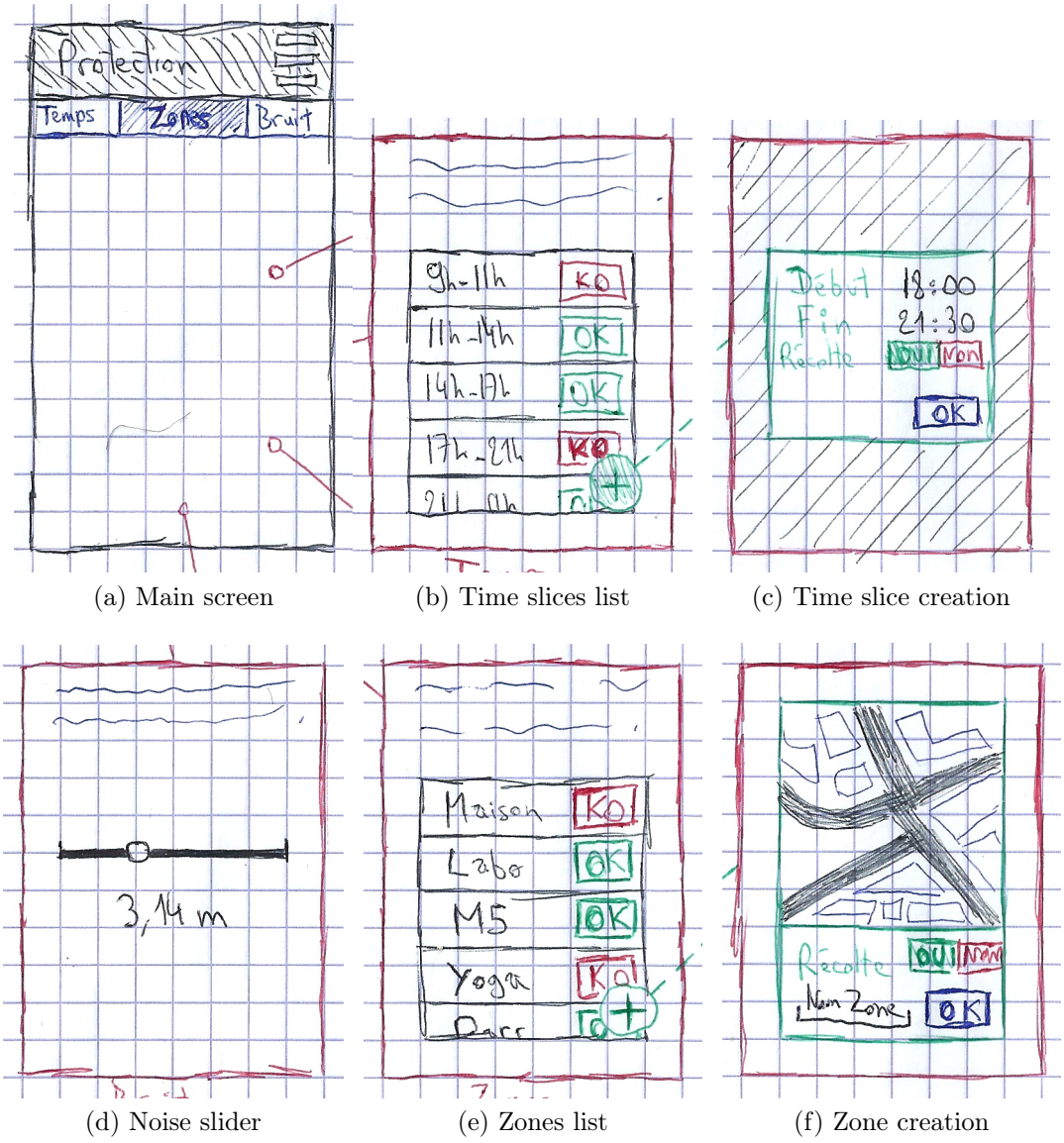


Figure A.1: Heatmap example generated with Haxe

Protection screen mock-ups



(a) Main screen (b) Time slices list (c) Time slice creation

(d) Noise slider (e) Zones list (f) Zone creation

Figure A.2: Protection component fragments drawings

Test scenario

This is an example of scenario used to simulate user interaction on test applications (all employed scenarios can be found at `https://github.com/rsain/Android-TPLs/tree/master/study`).

```
PACKAGE="ruben.soccerlab.acra"
ACTIVITY="ruben.soccerlab.acra.MainActivity"

# Launch app ("Launch app" entry in partition_info.csv file)
adb shell am start -n $PACKAGE/$ACTIVITY

# Wait few seconds to simulate user waiting
sleep 3

# Press button to crash the app
adb shell input tap 300 600

# Sleep some time (expecting that information is sent to the server)
sleep 3

# Close Android message about the crash
adb shell input tap 500 1300

# Sleep some time (expecting that information is sent to the server)
sleep 3
```

List of Figures

Bibliography

[1] Miguel Andrés, Nicolás Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. *ICDCIT, LNCS8956, Springer*, 2015.

[2] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Proceedings of IEEE Sixth International Symposium on Wearable Computing*, 2002.

[3] Sophie Cerf, Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, Robert Birke, Sara Bouchenak, Lydia Y. Chen, Nicolas Marchand, and Bogdan Robu. Pulp: Achieving privacy and utility trade-off in user mobility data. *36th IEEE International Symposium on Reliable Distributed Systems*, sep 2017.

[4] Sophie Cerf, Bogdan Robu, Nicolas Marchand, Sonia Mokhtar, and Sara Bouchenak. A control-theoretic approach for location privacy inmobile applications. *2nd IEEE Conference on Control Technology and Applications*, aug 2018.

[5] Cynthia Dwork. Differential privacy: A survey of results. *International Conference on Theory and Applications of Models of Computation*, 2008.

[6] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[7] Bugra Gedik and Ling Liu. Location privacy in mobile systems: A personalized anonymization model. *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, 2005.

[8] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, jan 2009.

[9] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. PrivÉ: Anonymous location-based queries in distributed mobile system. *Proceedings of the 16th International Conference on World Wide Web*, 2007.

[10] Google. Build for the next billion users. `https://developer.android.com/distribute/best-practices/develop/build-for-the-next-billion`, 2018.

[11] Thomas Hartmann, Francois Fouquet, Matthieu Jimenez, Romain Rouvoy, and Yves Le Traon. Analyzing complex data in motion at scale with temporal graphs. *Knowledge Engineering (SEKE'17)*, jul 2017.

[12] John Krumm. Inference attacks on location tracks. *Proceedings of the 5th international conference on pervasive computing*, 2007.

[13] Mohamed Maouche, Sonia Ben Mokhtar, and Sara Bouche. Ap-attack: A novel re-identification attack on mobility datasets. *Proceedings of the 14th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, nov 2017.

[14] Mohamed Maouche, Sonia Ben Mokhtar, and Sara Bouchenak. Hmc: Robust privacy protection of mobility data against multiple re-identification attacks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2018.

[15] Vincent Primault, Sonia Ben Mokhtar, Cédric Lauradoux, and Lionel Brunie. Differentially private location privacy in practice. *Third Workshop on Mobile Security Technologies (MoST)*, may 2014.

[16] Vincent Primault, Sonia Ben Mokhtar, Cédric Lauradoux, and Lionel Brunie. Time distortion anonymization for the publication of mobility data with high utility. *14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, aug 2015.

[17] Rubén Saborido, Foutse Khomh, and Yann-Gaël Guéhéneuc. Third-party libraries and their impact on android apps. *Draft, not proposed*, 2019.

[18] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information. *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1998.

[19] Statista. Number of smartphone users worldwide from 2014 to 2020 (in billions). `https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/`, 2015.

[20] Haoyu Wang, Yao Guo, Ziang Ma, and Xiangqun Che. Wukong: A scalable and accurate two-phase approach to android app clone detection. *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, jul 2015.

[21] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. Discovering personal gazetteers: An interactive clustering approach. *Proceedings of the 12th Annual ACM International Workshop on Ge-ographic Information Systems*, jan 2004.