TÉCNICO LISBOA

CROSS: loCation pROof techniqueS for consumer mobile applicationS

Gabriel Antunes Maia

Thesis to obtain the Master of Science Degree in

Information Systems and Software Engineering

Supervisor: Prof. Miguel Filipe Leitão Pardal

Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. Miguel Filipe Leitão Pardal
Member of the Committee: Prof. Alberto Manuel Ramos da Cunha

December 2019

Acknowledgments

This work would not have been possible without the guidance of my advisor Miguel Pardal. His help and coordination were essential throughout the development of this work, and I am very grateful for the opportunity to work with him.

I would like to extend my deepest gratitude to Rui Claro, for the notable effort of collecting Wi-Fi data throughout Lisbon, and for his help revising this document. I would also like to thank my friends and colleagues Henrique Santos and Pedro Carmo, for the ideas shared in the context of the SureThing project, and for their help with the evaluation and public presentations of my work.

I am also grateful to my friends Ricardo Farracho and Pedro Brás, with whom I shared impressions about the development of this work. Their patience and encouragement cannot be overestimated.

I would like to recognize the assistance of the volunteers who participated in the evaluation sessions, with special thanks to those part of the UnderLX community. Their names are too many to mention, nevertheless, their effort had a notable positive impact in the quality of the evaluation.

Last, but certainly not least, I would like to thank my family, particularly my parents, for supporting me throughout the years, so that I could achieve higher education in my favorite field.

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID) and through project with reference PTDC/CCI-COM/31440/2017 (SureThing).

Resumo

A omnipresença de *smartphones*, *wearables* e outros dispositivos móveis, associada com a crescente quantidade de infraestrutura de telecomunicações presente em ambientes urbanos, levaram à popularização de aplicações sensíveis à localização. Muitas destas aplicações não verificam a informação de localização que consomem, tornando-as vulneráveis a ataques de falsificação de localização. Os sistemas de prova de localização tentam resolver este problema, permitindo que os dispositivos interajam com recursos específicos de cada local, produzindo prova da sua presença em certo local a determinada hora.

Neste trabalho propomos o CROSS, um sistema que verifica informação de localização recorrendo a técnicas compatíveis com *smartphones* Android comuns. Apresentamos três estratégias para a produção de provas de localização com crescente resistência a ataques, duas das quais baseadas na tecnologia Wi-Fi e uma terceira baseada em interação física com quiosques eletrónicos ou dispositivos equivalentes. Estas estratégias foram desenvolvidas com a privacidade e segurança dos utilizadores em mente, minimizando a quantidade de ligações entre dispositivos. Foi desenvolvida uma aplicação protótipo para avaliar a viabilidade e fiabilidade da arquitetura e das estratégias de prova de localização. A aplicação permite recompensar utilizadores que completem circuitos turísticos, com provas de visita recolhidas ao longo do caminho. Escolhemos o turismo inteligente como contexto demonstrativo da utilidade e viabilidade das provas de localização. A avaliação realizada, que incluiu experiências com 30 utilizadores, com equipamentos diversos, indica que o sistema pode ser usado em cenários do mundo real, fornecendo garantias de segurança adequadas para o caso de uso em questão.

Palavras-chave: Prova de Localização, Adaptação ao Contexto, Segurança Móvel, Turismo Inteligente, Internet das Coisas.

Abstract

The ubiquitousness of smartphones, wearables and other mobile devices, coupled with the increasing amount of communications infrastructure present in urban environments, has led to the rise of location-aware applications. Many of these applications do not verify the location information they consume, making them vulnerable to location spoofing attacks. Location proof systems aim to solve this problem by allowing devices to interact with location-specific resources and issue proof that they have been at specific locations on specific times.

In this work we introduce CROSS, a system that performs location verification using techniques compatible with off-the-shelf Android smartphones. We present three strategies for the production of location proofs with increasing tamper-resistance, two of which are based on Wi-Fi and a third based on physical interaction with kiosk-like devices. Our techniques were designed with user privacy and security in mind, minimizing the amount of connections between devices. A prototype application was implemented to assess the feasibility and reliability of the architecture and location proof strategies. The application allows rewarding users who complete a touristic route, with proofs of visit collected along the way. We chose smart tourism as the demonstrative use case for the usefulness and feasibility of location proofs, in the context of a mobile application. Our evaluation, which included experiments with 30 users, using diverse equipment, showed that the system can be feasibly used in real-world scenarios, providing adequate security guarantees for the intended use case.

Keywords: Location Proof, Context-Awareness, Mobile Security, Smart Tourism, Internet of Things.

Contents

List of Tables

List of Figures

Acronyms

GIS Geographic Information Systems

GNSS Global Navigation Satellite Systems

POI Point of Interest

AP Access Point

BSSID Basic Service Set Identifier

SSID Service Set Identifier

HMAC Hash-based Message Authentication Code

TOTP Time-based One Time Password

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

TCP Transmission Control Protocol

UDP User Datagram Protocol

API Application Programming Interface

AR Augmented Reality

IDE Integrated Development Environment

IoT Internet of Things

JSON JavaScript Object Notation

REST REpresentational State Transfer

XML Extensible Markup Language

Chapter 1

Introduction

In the coming years, the amount of Internet-connected devices will increase by orders of magnitude [Int19, Str19]. Many of these devices are sensors and actuators that will be deployed in cities, which are gradually becoming smart cities. These devices will connect the physical and virtual worlds, constituting an Internet of Things (IoT) [ZBC+14]. Smartphones and wearables, which are now ubiquitous, will play an important role as user interfaces between people and the smart city infrastructure, paving the way to new context-aware applications.

Location information is one of the most frequently used context sources, with the first location-aware applications dating back to the end of the 1990s [BDR07]. In the Android ecosystem, about a quarter of the applications request access to location information [OA15]. Many of these applications do not verify the location information they consume, making them vulnerable to location spoofing attacks. *Location proof systems* focus on countering these attacks, by e.g. allowing devices to interact with location-specific resources and issue proof that they have been at specific locations on certain times.

One specific smart city application that could benefit from location proofs, for tourism and cultural promotion, can be called *smart tourism*. Smart tourism provides tourists with rich experiences supported by mobile technology and, overall, creates value from a business perspective [KK17, GSJ17]. According to Gretzel et al. [GSXK15], smart tourism can be considered a progression from traditional tourism that has greatly benefited from technological innovation, with applications for the discovery and reservation of accommodations, for example. These benefits are further enhanced by the widespread adoption of social media, the ubiquitousness of mobile devices and the development of an increasing number of smart city projects.

One possible form of smart tourism uses the personal devices of tourists to interact with existing or newly-added infrastructure in emblematic city locations, as illustrated in Figure 1.1. The ability to locate tourists in a reliable and verifiable way allows for the development of location-based smart

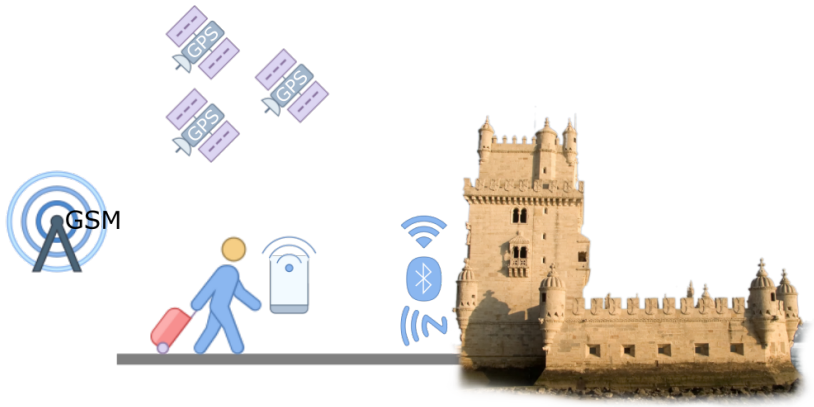


Figure 1.1: Representation of the technology available to smart tourism applications.

tourism applications and games. User location information is sensitive and preserving the privacy of the tourists is also a priority.

1.1 Motivation

Many mobile IoT applications use location context to provide their core functionality or to augment their capabilities [BDR07]. Only a minority of the currently available smart tourism applications uses the location context for non-informational purposes, for example, to reward users for visiting certain locations. Those which do, do not verify the location information they use, and can be deceived by different location spoofing techniques. Developing the means to validate location information is, therefore, of high importance.

Location proof systems differ from location systems in that they focus on countering location spoofing, by providing verifiable location information. The methods that can be used to produce location proofs depend on the available information sources and on the intended use cases. Wi-Fi can be used as infrastructure for location because most urban environments in populated areas tend to have many Wi-Fi networks. Some of these are for private or institutional use, while others are open for the general public to use. Nevertheless, the overwhelming majority of these networks announce their presence and can be detected using commodity smartphones.

This work was done in the context of the SureThing project. The SureThing project [FP18] allows for verifying the location of Internet-connected devices. The location component is very important for the design and enforcement of security policies suitable for the complex and dynamic reality of the IoT. This work illustrates the usefulness of the project with a valuable use case.

1.2 Contributions

Our main contribution is called CROSS (loCation pROof techniqueS for consumer mobile applicationS), a system that uses the Wi-Fi networks present in a predefined set of locations, to both detect the presence of the user in these locations, and to verify that the user is not spoofing his location. This information is used, in the prototype application, to ascertain whether the user completed one or more predefined tourism routes. System operators can then reward tourists who verifiably complete tourism circuits in the city with promotional gifts or coupons.

We evaluated different aspects of our system, including its security model, the accuracy of the location detection system, the ideal settings for location verification, and the effort required to maintain Wi-Fi Access Point databases and to add new locations to the system.

With this work, we show that location proofs can feasibly be used in the context of mobile applications, supported on hardware commonly found in consumer handhelds, using the infrastructure already present in cities and preserving the touristic experience.

1.3 Dissertation Outline

The remainder of this document is structured as follows. Chapter 2 presents concepts relevant to location systems, related work in the field of location systems, and a survey of existing smart tourism applications. Chapter 3 presents existing work in the field of location proof systems. Chapter 4 details the implementation of our solution, including its architecture, as well as the techniques and technologies used to implement the prototype. Chapter 5 presents the evaluation performed to assess the design of the solution and its performance. Finally, Chapter 6 presents the conclusion and future work.

Chapter 2

Background

Context-aware mobile applications play an important role in smart tourism, with location information being the most commonly used type of context [BDR07]. In Section 2.1, we survey the state of the art in location systems for mobile use. Privacy-preservation techniques will be discussed in Section 2.2, because tourism-related applications often acquire and process sensitive user information, including location information. Geographic information systems facilitate the development of location-aware solutions, and will be discussed in Section 2.3. Finally, in Section 2.4, we will summarize the current work in smart tourism applications.

2.1 Location in mobile applications

Location systems can provide *physical* and *symbolic* information. The former locates an object in absolute terms within a coordinate system, while the latter consists on abstract information about the position of an object. Symbolic location systems usually can only provide low-precision physical positions. Location systems can also be classified as *absolute*, where a shared reference is used for all located objects, and *relative*, where each object can have a different frame of reference. Absolute positions can typically be transformed into relative positions with ease, but the reverse can be complex, requiring the use of triangulation or trilateration and the knowledge of the absolute positions of other objects [HB01].

Location techniques can also be broadly divided into *geolocation* and *microlocation* techniques. The former focuses on determining geographic location, while the latter can determine the location of something within a limited space, such as an university campus, building or room, with typically much higher precision than geolocation techniques. Often, microlocation can be used for indoor location, i.e. in confined environments without a clear view of the sky and where precision is usually more important. A useful example of such an environment is a building with multiple floors, where

each floor is a separate physical space with different use assignments.

A summary of the considered location systems is shown in Table 2.1, showing which function they perform (scope), the platforms they target, the data sources they use, and whether they are *stand-alone*. We consider a location system to be *stand-alone* when it is able to locate the device where it is running without offloading parts of the computation to other systems. Similarly, sound recognition systems are *stand-alone* when they operate on the same device where sounds are captured without offloading the analysis.

System	Scope	Platform	Data sources	Stand-alone
Android Location APIs [Goob]	Geolocation	Android	GNSS, Wi-Fi, cell towers	Yes (with lower accuracy)
Apple Core Location [App]	Geolocation	iOS	GNSS, Wi-Fi, cell towers	Yes (with lower accuracy)
Google Indoor [Gooa]	Microlocation	Google Maps	Wi-Fi	No
HERE Indoor Positioning [HER]	Microlocation	Android, iOS	Wi-Fi, Bluetooth	Yes
SurroundSense [ACC09]	Microlocation	Nokia N95 (prototype)	Wi-Fi, sensors, camera	No
Baniukevic et al. [BJL13]	Microlocation	Android	Wi-Fi, Bluetooth	Yes (in thick-client mode)
SAIL [MSLK14]	Microlocation	Android, iOS	Wi-Fi, sensors	Yes (requires AP connection)
Pillos et al. [PAA+16]	Sound recognition	Android	Microphone	Yes
OtoSense [Oto]	Sound recognition	Not disclosed	Microphone, vibration sensor	No
AmbientSense [RFA+13]	Sound recognition	Android	Microphone	Yes

Table 2.1: Summary of the studied location systems.

2.1.1 Geolocation systems

Location context is relevant for many mobile applications. For this reason, the most widely used mobile operating systems, Android and iOS, both provide APIs that manage the access of applications to the Global Navigation Satellite Systems (GNSS) supported by most smartphones sold in the last decade [Can08, Ber10]. These systems can be used for stand-alone geolocation of a mobile device, but are of limited use in areas without line-of-sight to a sufficient number of satellites, including indoors and underground. Their accuracy can also vary widely, from 1 meter to over 50 meters, depending on the number and visibility of the satellites and characteristics of the receiver [Bau13, LGD+15].

Additional APIs made available by Google [Goob], Apple [App] and Microsoft [Mic] on Android, iOS and Windows, respectively, use their proprietary services to combine GNSS with additional

information, such as Wi-Fi access points and GSM[1] cell tower information, to determine the location of a device with better accuracy, lower power usage, or under poor sky visibility conditions. This additional information is kept in large databases, actively collected and maintained by the mentioned entities. However, these APIs often require the device to have an active Internet connection, and that users agree to privacy policies governing the access of these companies to user location data. These solutions are not stand-alone, as part of the computation is offloaded to online services.

2.1.2 Microlocation systems

Google provides an indoor location service [Gooa], but it is heavily coupled with Google Maps[2] and requires a partnership between location administrators and Google. The administrators authorize access to the physical location, Google collects measurements and stores them in a database. Another example of a indoor location service is HERE [HER], which offers a proprietary indoor positioning system which uses Wi-Fi and Bluetooth radios, and can be implemented on any location and application, subject to the terms of service and pricing tiers. Both Google Indoor Maps and HERE Indoor Positioning are commercial offerings, but there are also relevant research systems.

SurroundSense [ACC09] uses fingerprinting techniques to symbolically (or, in the words of the authors, *logically*) localize mobile phones, supported by Wi-Fi sensing, accelerometers, microphones and cameras. SurroundSense focuses on microlocation, useful for indoor location, and does not deal with geolocation. The authors of SurroundSense did not consider the energy trade-offs of their system. SurroundSense is not stand-alone, as it has a server component that performs most of the fingerprinting and raw data analysis.

Baniukevic et al. [BJL13] present three different techniques for indoor positioning which differ with regard to their power consumption, accuracy and co-existence properties of the two radio-frequency technologies used by this system: Wi-Fi and Bluetooth. An algorithm for efficient positioning of Bluetooth beacons is also presented. Power consumption and accuracy were studied for each of the techniques in order to attest the feasibility of the presented architectures. Two of the three architectures perform all position estimation on the client side, and can therefore be considered stand-alone microlocation techniques.

While the above systems require the presence of multiple beacons, e.g., Wi-Fi Access Points, in order to micro-localize a device, SAIL [MSLK14] is a dead-reckoning system that was designed to work with a single Wi-Fi Access Point in conjunction with the motion sensors of the mobile device. In this context, dead-reckoning is the use of motion sensors, to determine the device's displacement

[1] Global System for Mobile communications, the cellular communication standard, that popularized the use of mobile phones in Europe and around the world.

[2] An API for generic, uncoupled use of this service does not appear to exist.

relative to a previously determined location. SAIL uses Time-of-Flight-based distance estimation, using physical layer information, provided by Atheros wireless chipsets, to account for the multiple paths phenomenon, where a reflected wireless signal may be stronger than the signal corresponding to the direct path, due to e.g. obstacles in the way. In addition, SAIL presents techniques for reduction of errors in dead-reckoning based on the accelerometer, compass and gyroscope of the device. Accuracy of this system was thoroughly tested, but no results are shown in terms of energy consumption.

2.1.3 Ambient-sound systems

Sound recognition systems can be used to help identify symbolic locations as part of a more complete location system. They can be used to deduce that the user is at a train station by the noise of trains in the background, but on their own, they can not identify which train station or where it is located. These systems can also be used to perform activity recognition, i.e., recognize the activity of the user.

Pillos et al. [PAA+16] proposed an environmental sound recognition system for Android that runs in real-time, directly on the mobile device. Their system uses machine learning techniques for sound identification, and a sound detection algorithm to reduce power consumption. Their evaluation considered privacy concerns, accuracy and battery consumption. They concluded that their system is viable and has better accuracy (74.5%) than one other considered sound recognition system, OtoSense [Oto], while achieving about 80% longer battery life.

AmbientSense [RFA+13] is another sound recognition system, implemented as an Android application, supporting both on-device and server-based running modes. Evaluation used a set of 23 sound classes common in daily life. Compared to the system presented by Pillos et al., this work requires less computationally powerful hardware and consumes less energy, contributing to a longer battery life. Accuracy results are not directly comparable as different datasets were used during evaluation, but their system showed an accuracy higher than 80% for 12 sound classes and lower than 20% for 3 sound classes. AmbientSense authors concluded that due to the network communications overhead, the server-based running mode was not beneficial, unless more complex classification models, requiring more computational power, are used. The alternative mode runs entirely on-device and can be considered stand-alone.

2.2 Privacy-preservation techniques

Privacy is a primary concern when dealing with exact and certifiable user location information, as provided by location systems and location proof systems, because this information can be used to

track people. Beresford and Stajano [BS03] defined *location privacy* as a type of information privacy that consists in "the ability to prevent other parties from learning one's current or past location".

Location privacy has been an object of research since before location systems became ubiquitous with the rising popularity of smartphones. In 2002, Langheinrich presented pawS [Lan02], a system that lets users make privacy choices as location data is captured. In 2003, Myles et al. [MFD03] presented a system that, through the use of machine-readable privacy policies, controls the release of location data on a per-request basis. Beresford and Stajano [BS03] proposed a framework for privacy protection based on the use of pseudonyms that change periodically.

Fawaz and Chin [FS14] designed LP-Guardian, a location privacy protection framework for Android. It prevents user identification and can be deployed in practice without modifying other applications, while still providing useful location information. This solution decides whether to provide location information and if it should be anonymized on a per-application, per-location basis, requesting a decision from the user if preferences for a matching request have not been previously set. LP-Guardian can also provide applications with synthetic routes so those can calculate the traveled distance, for example, without accessing the true route coordinates. The evaluation performed by the authors, on Android 4.3, found LP-Guardian to be easy to deploy, to have acceptable energy consumption, and to cause a tolerable loss in application functionality. Because LP-Guardian requires the Android operating system to be modified, it does not work with the smartphones currently in the market. The techniques used for privacy protection also appear to be incompatible with most location proof systems described above, as they would interfere with their ability to obtain their true location in the first place.

Agadakos et al. [AHD+16] proposed the Icelus system, which can locate users and model their movement through the use of IoT devices and smart environments. The paper details concrete measures for preserving privacy when parts of the system are hosted by third parties: third-party hubs can only learn the distance between devices, and not the positions of the devices. This is ensured through the use of a custom sub-protocol for communication between the hubs and the entities that request location proofs. This sub-protocol uses *secure multi-party computation*, *additively homomorphic encryption* and *additive blinding*.

2.3 Geographical Information Systems

Geographic information systems (GIS) can be used in the context of a smart tourism application as the means to locate and manage information about the points of interest and tourism routes. Web mapping software such as Google Maps, Bing Maps and OpenStreetMap, and satellite imagery software, of which Google Earth is an example, are widely-known types of GIS. These systems are

used by tourists to plan their trips and by businesses to promote and plan touristic offerings [KK17].

Web mapping systems sometimes feature user-contributed or crowdsourced [Bra08] information, with users being able to submit corrections or new points of interest. Crowdmapping corresponds to the use of crowdsourcing to maintain up-to-date geographical information and other mapping-related information [SE15]. OpenStreetMap[3] is notable in this regard because it is a collaborative project built entirely on crowdsourced data. Often, web mapping systems expose APIs[4] that allow their integration in custom applications. MapBox[5] is one example of a web mapping system whose primary focus is providing custom maps for embedding in other applications.

2.4 Smart Tourism applications

We performed a brief analysis of over 50 popular smart tourism applications and identified common features that allow us to classify and categorize them as informational, georeferenced or location-specific.

Informational applications show information of interest to tourists, e.g. attractions, monuments, hospitality, routes and activities, featuring the whole world or specific countries, regions or cities. The Culture Trip and TripAdvisor are examples of such applications. Informational applications frequently feature public-domain and crowdsourced/user-contributed information, including ratings and reviews of the points of interest. Sometimes the informational aspect is included as part of a larger application, as is the case of the "travel tips" feature in the Booking.com application.

Informational applications often handle georeferenced information, that is coupled with geolocation systems to let tourists know about points of interest in their vicinity (e.g. with a map showing the current location along with the nearby points of interest). We consider those to be *georeferenced applications*. Georeferencing features are especially important when the application contains data for the whole world, because they allow for information filtering without requiring that the users manually specify their location. Georeferenced applications, in more general terms, are those that take into account the location context of the user, to offer different content or functionality depending on their current and past location, but do not include microlocation information in that context. Notable applications on this category include: FourSquare, which provides personalized destination recommendations based on the location and history of the user; Google Trips, which automatically creates trip plans based on the location of the user; and Sygic, another location-aware trip planner and tourism guide.

Location-specific applications show information about a single tourist destination or venue, in

[3] `https://www.openstreetmap.org/`
[4] Application Programming Interfaces
[5] `https://www.mapbox.com/`

contrast with georeferenced applications, which generally cover a large geographical region or even the whole world. Some employ microlocation techniques. These applications are often developed and operated by the venue owner or responsible entity, and are sometimes marked as being the official one for that venue. Some examples of location-specific applications include "My Visit to the Louvre" (official Musée du Louvre application, with indoor location features), the official Portuguese Coach Museum application, and the Disneyland application.

There are many other types of applications that, despite not having tourism as their main use case, can be considered smart tourism applications, because they enable mobile technology to enrich the experience of tourists - thus perfectly fitting the smart tourism definition we presented previously. These include mapping and navigation applications, transit applications, ride-sharing and rent-a-vehicle applications, and language translation applications[6]. Many of these make heavy use of the geolocation context, and could benefit from location proof systems. For instance, rent-a-vehicle applications could use location proofs to ensure users do not leave the area of operation of the service, and to defend against manipulation of the billing system in pay-per-distance schemes.

2.4.1 Microlocation applications

Applications with microlocation functionality are a much smaller subset of the smart tourism applications we analysed. Google Maps supports indoor location through the same service we described in 2.1 [Gooa]. Since November 2018, this service is also used in the "Find My Device" feature available for Android smartphones. Besides generic mapping and navigation, other applications with microlocation features are typically location-specific, as is the case of the aforementioned "My Visit to the Louvre" application.

According to the user reviews of applications with microlocation features, these are often inaccurate and are perceived not to add much value to the core application functionality. Without the ability to test these applications in a real-world setting it is impossible for us to understand whether this sentiment is due to deficiencies of the location technology, because of poor user interface and interaction design, or due to incorrectly identified requirements.

2.4.2 Applications with reward schemes

Of the applications we analysed, none made use of the location context to reward users based on their location history. Stepping outside of the context of smart tourism, geocaching is a location-based recreational activity in which navigational and location techniques, including those supported by GNSS, are used to play a form of "hide and seek"; mobile applications exist to assist the practice

[6]Google Translate and Microsoft Translator are examples of machine translation applications available for mobile platforms.

of this activity. Beyond geocaching, Ingress and Pokémon Go are two examples of location-based, augmented reality (AR) games that were popularized in recent years. We believe these games and applications, while providing users with rewards based on their location, can not be classified as smart tourism applications as clearly as the others we analysed. Their users do not necessarily follow well-established tourist routes or visit particular points of interest, and often travel in ways that are most advantageous to the gameplay but serve no touristic purpose. These applications can often relax their requirements for location verification, as the nature of the rewards ensures that any innacuracies or spoofing will usually only have consequences inside of the game itself.

2.5 Summary

The development of location-aware smart tourism mobile applications is closely related to work in geographic information systems and location systems, which we analyzed in this chapter. We also looked into privacy-preservation techniques, which are relevant when dealing with user location information, but none of them are easily applicable to the use case of location proofs for smart tourism.

We briefly studied existing smart tourism applications, that we classified as informational applications, georeferenced applications, and location-specific applications. We could not identify any smart tourism application that contained reward schemes, or used location proof techniques. Other applications, such as location-based games, contain reward schemes, but cannot be clearly considered smart tourism applications. In the next chapter, we will look into existing works in the field of location proof systems, which could be used to verify the location information consumed by mobile applications.

Chapter 3

Location Proof Systems

The location information provided by location systems, such as those studied in Section 2.1, is vulnerable to location spoofing attacks, where the systems, or the consumers of their results, are led to believe they are at a location different from their actual one. Location proof systems focus on countering location spoofing, going beyond locating the user, by also proving that location in a verifiable way. We are mainly interested in systems that can be used in the context of a mobile application.

Use cases for location proof systems include reward schemes, such as discount programs for regular store customers, and location-based authentication schemes, for example, a smart door lock that only grants access when the homeowners are in its vicinity, or a fraud detection system that checks whether the holder of a credit card is present at the store when an in-person purchase is made.

3.1 Existing systems

CREPUSCOLO [CCCP13] uses strategically placed *Token Providers* and neighboring devices to provide location verification, even in areas with reduced amounts of users. Privacy is preserved through the use of periodically changing pseudonyms. In contrast with neighbor-based location proof systems APPLAUS [ZC11] and LINK [TCB12], CREPUSCOLO can protect against not just simple collusion attacks but also wormhole attacks, where packets are tunneled by an attacker from one physical place to another.

The SureThing system [FP18] allows devices to produce and validate location certificates, to make proof of their location and to reliably verify the location of other devices. Like CREPUSCOLO, SureThing uses neighboring devices as witnesses, together with the geographic location obtained by each device, e.g. using GNSS. A central component, the *verifier*, is responsible for certifying proofs.

To prevent collusion, the system takes advantage of the diversity of witnesses and of the concept of *redundancy* (multiple witnesses) and *decay* (proofs from the same witness lose their value as they are used to testify user presence).

LINK [TCB12] works similarly to SureThing, using neighboring devices reachable over short-range wireless technologies, such as Bluetooth, as witnesses. This system also associates a trust score with each device. As with SureThing, a central component is responsible for the decision process that certifies claims. In LINK, this process takes into account historical data and trust score trends to detect colluding users. Unlike SureThing, LINK does not make use of witness diversity: while each user/device is still expected to have the means to locate themselves (e.g. using GNSS), they are not expected to collect different types of location proofs from the environment, but from LINK neighbors only (it assumes users are not alone very often).

PROPS [GKRT14] is another neighbor-based location proof system with a focus on privacy preservation. In this system, the use of *zero-knowledge proofs* and *commitment schemes* allows prover and witnesses to remain anonymous and unlinkable. Witness locations are never disclosed, as their presence in the vicinity of the prover is checked using a proximity testing protocol. The location of the prover is not disclosed to the witnesses, as it is hashed before being endorsed by the witnesses. PROPS authors implemented an Android prototype application and carried out experiments using two devices connected to the same network.

Khan et al. [KZHH14] presented OTIT, a model that addresses secure *location provenance*, that is, the ability to prove a history of locations and not just the presence at an individual location. Location history is crucial in certain scenarios; for example, when analyzing the supply chain of a product, it is often important to prove through which countries it passed, or not; whether it went through the expected border checkpoints, and so on. OTIT authors presented different approaches, proven formally using their model, that can be used for chronological order preservation of location records. The OTIT model introduces requirements for location provenance systems and can be used as a benchmark framework for this type of systems.

The Icelus system [AHD+16], already mentioned in Section 2.2, uses the IoT to estimate the location of a user, for authentication purposes. The accuracy and efficiency of the system were evaluated and compared with smartphone-only approaches, with Icelus presenting lower false acceptance and false rejection rates. The researchers concluded that their approach exhibited a low impact on the performance of the tested IoT devices, with effectiveness bound to improve as the number of deployed IoT devices increases.

3.2 Discussion

Most location proof systems focus on providing strong tamper-resistance, with many adopting witness-based strategies, where peer-to-peer communication is used between the prover and witnesses. The communication, task scheduling and power management frameworks of current consumer-oriented mobile operating systems (Android and iOS) are heavily oriented towards a client-server model,[1] discouraging the use of peer-to-peer solutions.

Witness-based approaches have the drawback of requiring a minimum amount of users at each location where one may want to prove their presence. Some of these systems also have user privacy implications, as the peer-to-peer communication will disclose, at the very minimum, hardware identifiers such as MAC addresses. Randomization and pseudonymization of these identifiers is possible, but requires the collaboration of the hardware and operating system of the devices.

CREPUSCOLO [CCCP13] introduces the concept of *Token Providers*, which, in addition to providing resiliency against collusion attacks, allow the system to operate when there are no witnesses in the vicinity of the prover. Because these Token Providers must be deployed specifically for use with CREPUSCOLO, the cost and complexity of setting up this system is increased.

Other systems, such as Icelus [AHD+16], rely even more on the deployment of infrastructure, or customization of existing devices, to verify location information. This reduces the set of environments where such systems can operate, as well as the size of the potential user base.

The location proof techniques which we will introduce in the next chapter, present an alternative to witness-based approaches, using existing, unmodified infrastructure where possible. This helps reduce the costs, time and permits associated with the acquisition and installation of infrastructure. Our techniques work on common off-the-shelf, unmodified smartphones, ensuring their feasibility in real world scenarios and their ability to reach a large number of users. We will use smart tourism as a demonstrative use case, notwithstanding other possible use cases for these techniques.

[1] Applications are often prevented from keeping ongoing connections during standby. Firewall traversal and graceful handling of connectivity changes are complex in a mobile environment. Officially supported push notification systems in Android and iOS are centralized (Firebase Cloud Messaging and Apple Push Notification Service, respectively).

Chapter 4

Implementation

We developed CROSS, a mobile application, for smartphones, that rewards tourists for completing tourism circuits and visiting a set of points of interest. This application demonstrates the usefulness of location proofs in the context of smart tourism, using a variety of location proof techniques. In this Chapter we will present and discuss the implementation of this solution.

In Section 4.1, we discuss the requirements identified for this application and the assumptions made in its development. In Section 4.2, we present an overview of the architecture of the solution, explaining the main components, their purpose and how they interact. The location proof techniques employed in our system are presented in Section 4.3. In Section 4.4, we mention the software and platforms supporting our solution, explaining the reasoning behind our choices. An example of the operation of the system from the point of view of the users, from configuration to tourist usage, is given in Section 4.5. Finally, in Section 4.6, we summarize the key aspects of our solution.

4.1 Requirements

A use case diagram representing how system operators, POI officials and tourists interact with our solution is shown in Figure 4.1. We assume that the *system operators* (e.g. businesses in the tourism industry) define emblematic city locations as points of interest, circuits and corresponding rewards. In other words, our solution requires that points of interest and tourism circuits be manually identified. Points of interest may be owned, administered or operated by entities, which are not necessarily *system operators* and which might not be affiliated with them. We will call these entities *Point of Interest officials*, or simply *POI officials*. *Tourists* complete routes by visiting points of interest. For doing so, they receive rewards, or tokens that can be exchanged for rewards, at the end of each route.

The intended operation of the system from the point of view of the user is illustrated in Figure 4.2.

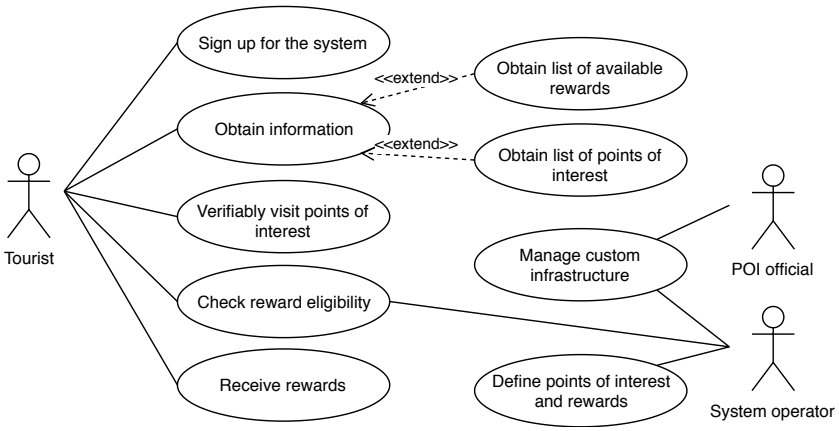


Figure 4.1: Use cases of the proposed solution and associated entities.

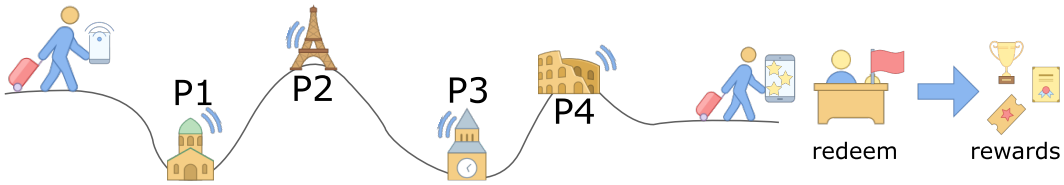


Figure 4.2: User flow throughout a tourism route with four points of interest.

Tourists, equipped with their personal smartphones, visit different points of interest, represented in the figure as P1 through P4. At the end of the route, the user is eligible to receive rewards, which can be redeemed with the cooperation of the system operators.

Our system was developed for environments with Wi-Fi networks. As mentioned in Section 2.1 and Chapter 3, Wi-Fi can be used to locate users, with different degrees of accuracy, and prove their location. The amount of networks required by our system depends on the location proof strategy one intends to use at each location. Some strategies can work with a single Wi-Fi Access Point.

To support areas with insufficient preexisting infrastructure and to increase the strength of proofs in locations with high-value rewards, our solution allows for custom infrastructure to be installed and used in the production of location proofs. The installation and maintenance of this infrastructure on different points of interest requires the collaboration of the respective POI officials.

It is desirable that the application be able to operate without a constantly available Internet connection, as tourists may visit areas without sufficient cell tower coverage for mobile data to work reliably, or they may need to avoid roaming charges altogether and rely on public Wi-Fi access points.

Because the aforementioned rewards may have a tangible value associated with them, one functional requirement is that rewards should only be given to people who are actually eligible to receive them. This is where location proofs will be useful. It is important to note that tourists are not the only entities who may have an interest in exploiting the system: POI officials may try to manipulate location proofs to make their points of interest appear more popular than they are (for example, to

manipulate recommendation systems), or less popular (e.g. to aid in tax evasion schemes and to qualify for undue subsidies).

A related non-functional requirement is that rewards should be assigned timely; in practice, this means that rewards should be assigned as soon as users finish the associated route or visit, as represented in Figure 4.2. As an example, the system will not be very useful if tourists only receive their rewards two weeks after they have left the city.

One functional requirement is that system operators be able to identify which users were rewarded with which rewards. This is necessary for some types of rewards, for example, physical prizes. The system may be extended such that other types of rewards, such as vouchers, coupon codes and digital licenses, are automatically assigned without intervention from the system operators.

4.2 Architecture

The CROSS architecture has four main components: client application, server accessed through API, Wi-Fi Access Point (for proof strategy described in 4.3.3), and Kiosk (for proof strategy described in 4.3.4).

The client and server constitute a traditional client-server model. The client, whose internal structure is represented on the lower side of Figure 4.3, is the application that is meant to run on the mobile phones of the tourists. The client interacts with kiosks and Wi-Fi Access Points to produce location proofs. The server, represented on the upper side of the figure, corresponds to the central component that is responsible for assigning rewards. Each user has an account in the system, with credentials that allow for their identification and authentication in the service.

In our solution, one entity (the system operators) has the complete power of decision over the workings of the system, making a centralized model more appropriate. A more decentralized model could involve peer-to-peer communication, which has some drawbacks, as mentioned in Section 3.2.

The server provides a service that lets clients fetch information about the available tourism routes, points of interest and possible rewards. This set of information, which is not user-specific and will typically not change frequently (more than once per day), will be hereinafter referred to as *catalog*. The service is also responsible for authenticating users in the system and, upon successful authentication, providing information specific to each account, such as past location history and the awards received. We will refer to this information as *user information*.

On the server side, a database is used to access and manage the catalog, information necessary for location proof verification, and user information. The *location proof verifier* component on the server is responsible for verifying the validity of location proofs received from clients by the API request handler. Rewards are assigned by the *reward assigner*, based on the valid location proofs

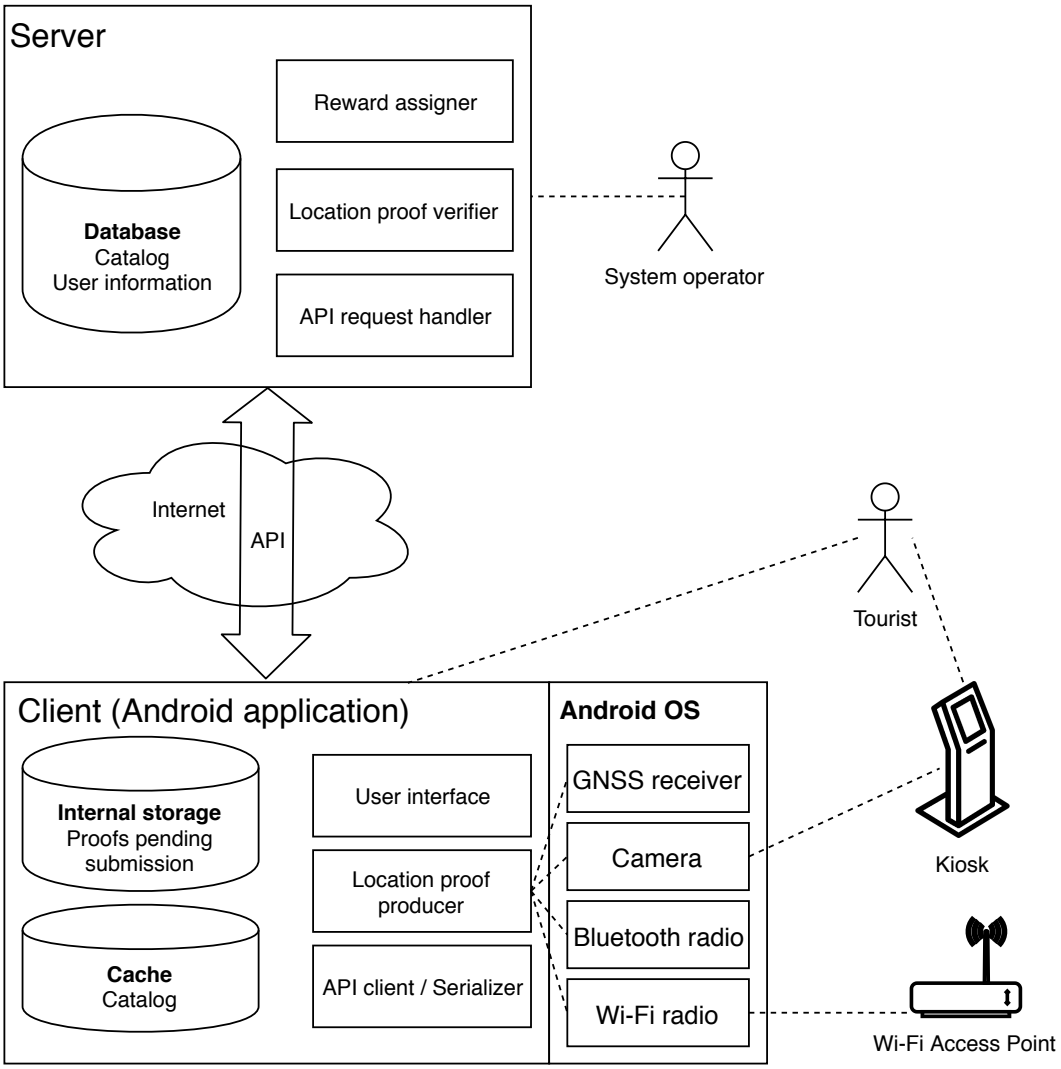


Figure 4.3: Overview of the architecture of the developed solution.

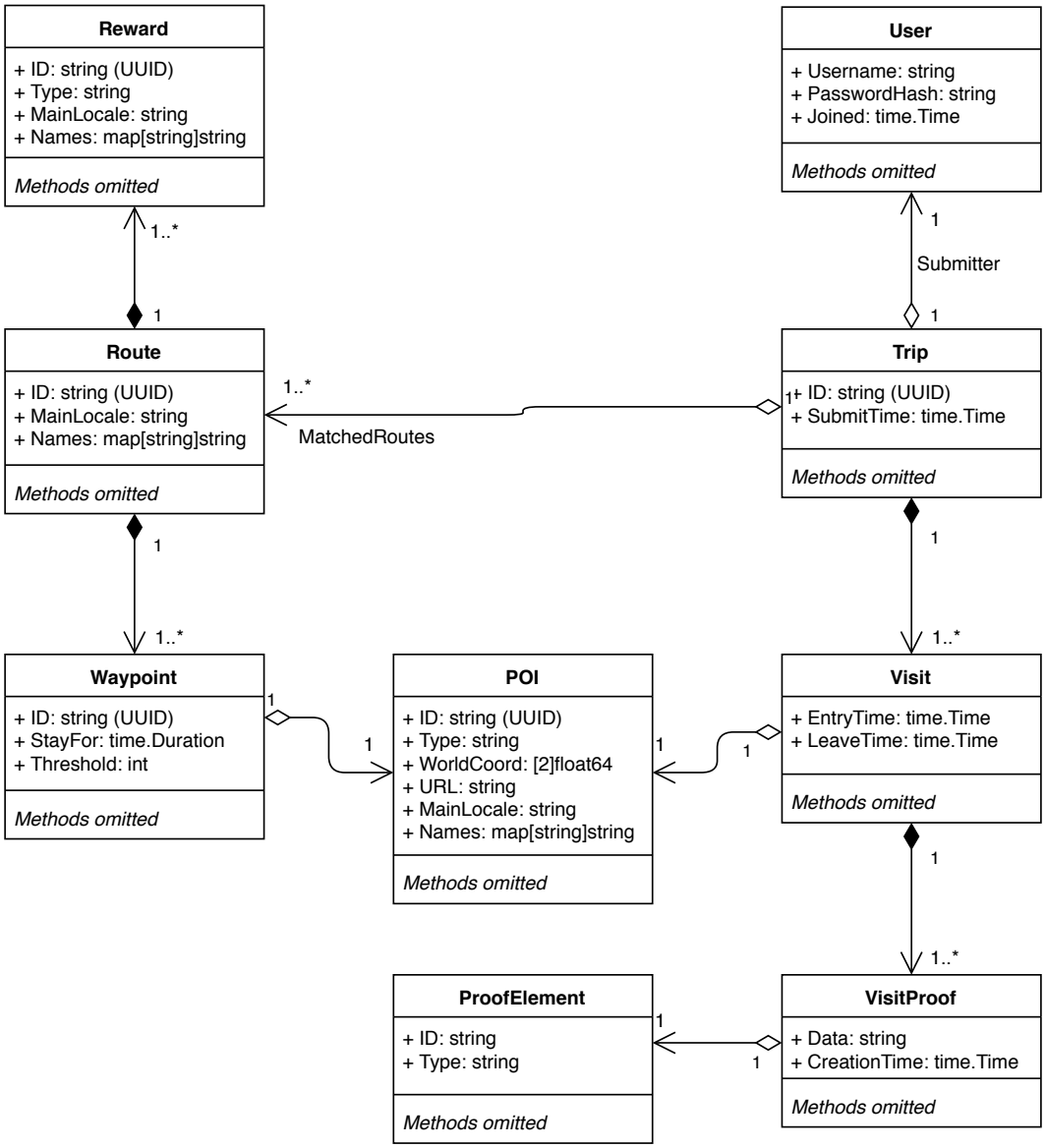


Figure 4.4: Diagram of the entities used in the server domain logic.

received, the information in the catalog and the user information relative to each client.

A simplified diagram of the entities used in the server implementation of the domain logic, and how they relate to each other, is shown in Figure 4.4. Of the shown entities, *POI*, *Waypoint*, *Route*, *Reward* and *ProofElement* must be configured by the *system operators* when setting up the system. *ProofElement* includes the Wi-Fi Access Points, Kiosks and other elements used as part of location proofs, that will be described in Section 4.3, and which the server must know about.

The service is meant to be available over the Internet; the server component is meant to be managed by the system operators. The client is an application for smartphones, that will fetch the catalog and, upon authentication, the user information from the server. This application is meant to be used by tourists as they visit different points of interest. The client is responsible for checking whether the user is near a point of interest, producing location proofs and sending them to the server;

it also lets users view the available and earned rewards. It caches information and stores location proofs for delayed submission, if the device is not online.

4.2.1 Communication protocols

Communication between the client and the server takes place over a HTTPS REST API. The client is able to work without a constant connection to the server. For this purpose, it caches the catalog, some user information, and location proofs for opportunistic submission.

HTTP is a web application protocol supported on reliable TCP connections. It is well understood and widely supported by most commonly used programming languages, and a large number of libraries exist to support its use. The use of a connectionless custom protocol based on unreliable UDP is not adequate to the type of communication between the server and the client, which we want to be as reliable as possible. In all foreseeable communication scenarios between the client and the server, the delay caused by TCP retransmissions and multiple connection attempts is acceptable. On the client side, we decided to set an upper bound on communication delays (timeout) of ten seconds, as regular communications are expected to take, at most, a few seconds, as is typical in mobile applications. The use of a TCP-based protocol other than HTTP was considered, but we could not identify any advantages of this option over a well-organized HTTP API.

REST (REpresentational State Transfer) is an architectural style for web services, defining a set of constraints restricting the way servers respond to requests. In practice, these constraints help meet some of the requirements for our solution; for example, the application of REST principles leads to a stateless protocol (from the application layer perspective). In stateless protocols, there is no concept of sessions and each request contains all the information necessary to be processed. This is useful in contexts where requests may need to be reattempted multiple times. Stateless protocols also simplify testing.

REST does not define a message format. We use JSON[1], a widely supported text-based data format that is a lighter alternative to XML, in terms of encoding and decoding complexity and message size. The server and client can be made to use other message formats, such as MessagePack, with minimal changes.

4.2.2 Security and privacy considerations

In our client-server model, the server is generally considered a trusted entity, while the client is untrusted. This means that all information provided by the client is validated by the server, including, notably, the location proofs. Despite the server being a trusted party, clients should still verify that

[1] JavaScript Object Notation

the information they receive matches the format they expect for each request, as a preventive measure against communication failures (e.g. due to the server being under maintenance). The *API client* component of the client is responsible for handling this verification.

The use of HTTPS (HTTP over TLS[2]) provides basic security on the connection including integrity checks and encryption. If care is taken to use and verify both client and server certificates, HTTPS can also provide authentication based on public-key cryptography. This method is not commonly used as it requires additional key and certificate management by the client, special web server configuration, and additional code on the client side. For these reasons, we decided against using this type of client authentication.

For authentication, we use an alternative method that is simpler, more commonly used and relies on TLS without involving client certificates: upon successful authentication on the web service, clients are provided with a *API token* that the API client component will send on subsequent requests, and which will be verified by the API request handler on the server, thereby authenticating the client. To authenticate the server, *certificate pinning* could be used, where client-accepted certificates are limited to a specific public key or to those signed by another trusted certificate. The pinned certificate, or the information specifying which certificates to accept, would be bundled with the application. This authenticates the server as long as the private key of its certificate is not compromised, because even if an impostor server manages to obtain a valid certificate for the FQDN[3] of the server, it will not match the pinned certificate.

As mentioned in Section 2.2, user privacy is of high importance, especially when dealing with location information. The use of HTTPS protects the privacy of the users from the perspective of an observer, that is able to capture all network traffic, but it does not deny the system operators access to the location history of a user. One possible mitigation would be to generate random pseudonyms to identify users, avoiding the propagation of personal information throughout the server database. However, to allow for the identification of prize winners, these pseudonyms must be associated with personal information at some point. One idea, which we did not have the opportunity to explore, is to asymmetrically encrypt all personally identifiable information on the client side, using private keys protected by the password of the users. This way, only the respective clients/users are able to decrypt it upon being rewarded, if they choose to do so. The system operators can not obtain this personal information on their own, but it is still stored encrypted on the server database for the convenience of the users.

[2]Transport Layer Security
[3]Fully qualified domain name

4.3 Location proof techniques

We propose three different strategies for location proof production and verification, with increasingly stronger guarantees. The first strategy, *scavenging*, relies solely on existing Wi-Fi networks, deployed by third-parties. The second strategy, *TOTP* (Time-based One-time Password), relies on Wi-Fi infrastructure deployed and configured specifically for use with CROSS. The third strategy, *Kiosk*, requires users to physically interact with an electronic kiosk booth.

Our first two strategies take advantage of existing infrastructure that smartphone radios can interact with, to fingerprint a location whenever a client needs to produce a proof of location. In practice, such infrastructure amounts to any Wi-Fi networks that may already be present in the environment; fingerprinting is performed by extracting device identifiers from network/beacon scan results. Of the currently relevant mobile operating systems, only Android provides access to Wi-Fi scan results.

4.3.1 Confidence score

Upon submission to the server, location proofs are associated with a *confidence score*, calculated by the server. This value is calculated differently by each proof strategy, and is not comparable across strategies. In the case of rewards for completion of routes encompassing multiple locations, the *confidence score* is calculated per location. The user is only considered to have completed a route if his visits to all locations have a score above the minimum acceptable threshold for that location in that particular route.

System operators are expected to configure the system such that higher value rewards require stronger proofs. This may be done by using proof strategies that provide stronger guarantees and/or increasing the acceptable score thresholds for a given strategy.

Protection against travel time inconsistencies

The *confidence score* suffers a penalty when the user is changing location at a pace inconsistent with the means of movement assumed to be at his disposal. This detection of travel time inconsistencies is a small, additional measure to increase the effort necessary to fool the system.

When the user moves from predefined location **A** to predefined location **B**, the $distance(A, B)$ in meters between locations is computed using the Haversine formula on the geographical coordinates of the two locations, previously configured in the system. $time(A, B)$ is the difference, in seconds, between the user leaving location **A** and entering location **B**, as registered by the client application. If the locations are more than 500 meters apart, the score associated with the visit to location **B** is multiplied by the result of formula 4.1, to penalize the scores of inconsistent visit proofs.

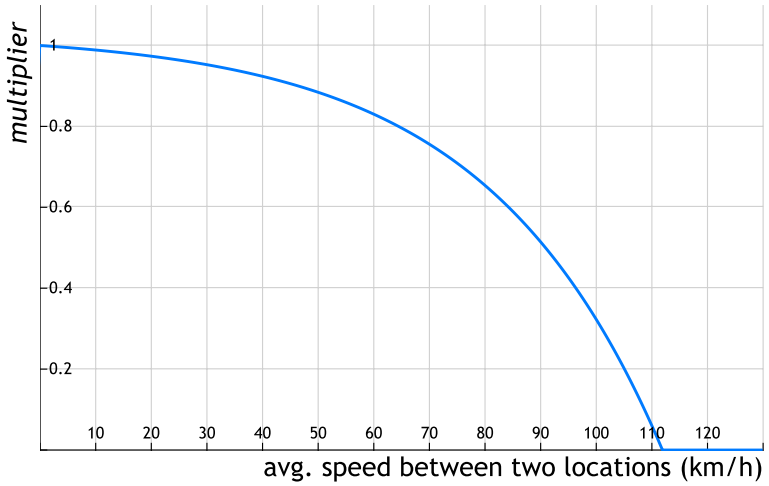


Figure 4.5: Graph showing the $multiplier$ in function of the average speed between two locations, as registered by the client application.

$$multiplier = max(0, 1.03 - 0.03 \times e^{\left(0.11376 \times \frac{distance(A,B)}{time(A,B)}\right)}) \tag{4.1}$$

Because Wi-Fi networks can have a long range, and because a high polling frequency is not attainable, when locations **A** and **B** are sufficiently close, the application will often register the movement from one location to another as being much faster than it actually was. At a limit, $time(A, B) = 0$ and the speed is undefined. Therefore, the system ignores travel time inconsistencies for locations less than 500 meters apart, a value selected based on the average network range we observed in a urban environment.

Formula 4.1, graphed in Figure 4.5 as a function of the speed in km/h, was obtained using exponential regression from a manually selected set of values. It should be adjusted, or a different formula could be used entirely, to reflect the means of transportation available to the user. For example, in a context where locations are in two far apart cities connected by high-speed train, the formula must be adjusted to consider much higher average speeds as legitimate. This protection applies to all of the location proof strategies detailed below.

4.3.2 Scavenging strategy

The idea for this strategy is to harness the large amount of Wi-Fi networks installed by third parties in urban environments. These networks may appear and disappear at any time. In this strategy, represented in Figure 4.6, location proofs are produced simply by storing Wi-Fi scan results with associated timestamps. These results are then submitted as part of the trip log.

On the server side, the set of Wi-Fi networks present in the scan results is compared with a list of known networks for each location. This list is curated by the system operators. The server periodically

analyzes past location proofs to suggest the addition and removal of certain Wi-Fi networks from the list. In this strategy, the *confidence score* is the fraction of client-presented networks over the total number of server-known networks.

Identifying the ideal confidence score threshold can be complex and will usually require on-field experiments. In general, the weaker the average signal strength of the networks at a location, the lower the threshold must be, in order to consider a fair amount of visits as verified. This is because the weaker the signal of each individual network is, the less likely it is for it to be detected and registered in the trip log. Therefore, this strategy provides stronger guarantees when more networks with strong signal levels are present at a location.

The main advantage of the scavenging strategy is its simplicity and reduced setup cost, as it just uses existing infrastructure. However, it is also the strategy that provides the weakest guarantees: as soon as the list of networks at a certain location is known, an attacker can forge trip logs.

4.3.3 TOTP strategy

This strategy allows for stronger proofs by deploying a customized Wi-Fi access point that is dynamically changing the broadcast SSID[4], as depicted in Figure 4.7. The SSID is used as a low-bandwidth, unidirectional communication channel to transmit a changing value: the digits in the network name.

This strategy is compliant with Wi-Fi standards [IEE] and compatible with existing devices. Note that the device observes the changing SSID values and does not need to connect to the network. Networks used in this strategy can be broadcast by existing APs, without disrupting the existing services, as typically each AP can support more than one Wi-Fi network.

Time-based SSID setting

The SSID should change in a way that is unpredictable to an observer, but which can be verified by the server. We achieve this by including a Time-based One-Time Password (TOTP), similar to the

[4]Service Set Identifier, the user-facing name for a Wi-Fi network

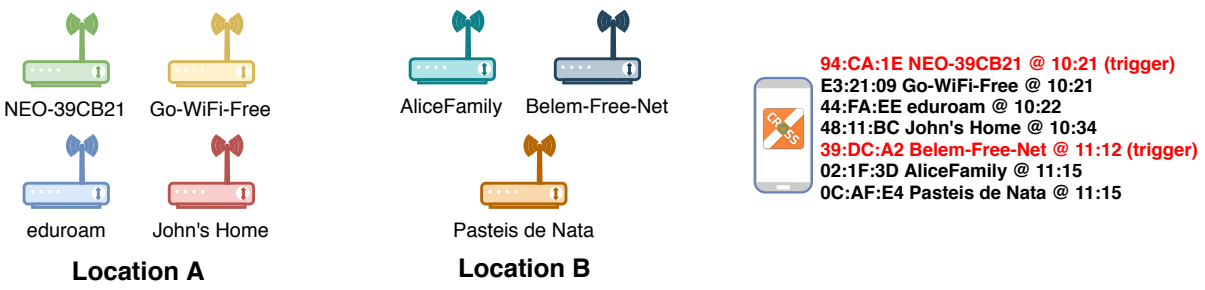


Figure 4.6: Representation of the networks and logs in a visit to two locations, A and B, where the scavenging strategy is used. Locations are sensed using trigger networks.

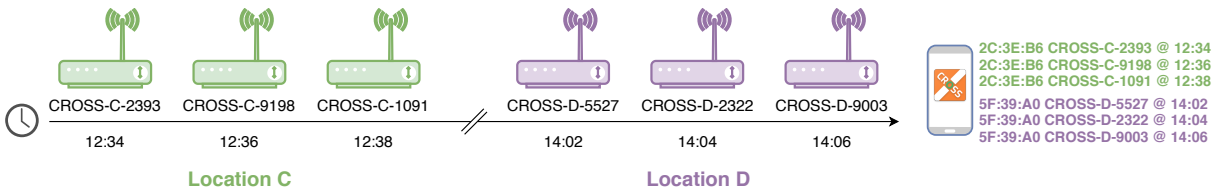


Figure 4.7: Representation of the networks and logged information in a visit to two locations, C and D, where the TOTP strategy is used. There is one AP at each location.

proposed in RFC 6238 [MMPR11], in the SSID value. Only the Wi-Fi Access Point (AP) and the CROSS server know the TOTP secret, to produce and validate OTPs, respectively. Each AP should use a different secret key, and only the server should know the keys used by all APs. The APs and server must have synchronized clocks with minute granularity, but both components do not need to communicate, which means APs can function as stand-alone beacons in locations without Internet access.

Our solution uses a carefully selected time-step size and hash algorithm, that are different from those recommended in RFC 6238, as our use case is different from the typical TOTP use case where the one-time password acts as a second authentication factor. We use a time-step size of 120 seconds, sufficient to provide enough resolution during proof verification, while still fitting within the constraints of most Wi-Fi Stations when it comes to updating scan results. We chose SHA-512 HMAC as the TOTP hash algorithm, with keys as long as the HMAC output, instead of the typically used SHA-1 HMAC. This allows the use of longer keys for additional security. This algorithm was selected to ensure that it is computationally complex to perform a key-recovery attack, i.e., infer the secret TOTP key by continuously observing the different SSIDs assumed by the AP. To the best of our knowledge, such an attack against SHA-512 HMAC is yet to be achieved [DEM15], unlike HMAC using weaker hash algorithms [CY06].

Proof collection and validation

Clients are programmed to log all the different SSIDs a Wi-Fi network assumes during their visit to a location, along with the observation timestamp. Clients do not know whether each Wi-Fi network is part of the infrastructure for this strategy, as that is irrelevant to how they collect proofs; only the server needs to know this, to select the correct proof validation strategy. In other words, as far as the client implementation is concerned, the scavenging strategy and the TOTP strategy are the same.

The TOTP strategy, unlike the scavenging one, allows for attesting not just that the user was present at a certain location, but also that he did so at a certain point in time. Therefore, this strategy can verify the visit duration, and helps establish *location provenance*, as defined by Khan et

al. [KZHH14].

In this strategy, the *confidence score* corresponds to the fraction of visit time that could be verified, in relation to the total time the client claims to have been present at the location. For example, if the client claims to have been present at a location for 20 minutes, but only 7 OTPs could be verified, corresponding to a total of 14 minutes within the claimed 20 minutes period, the confidence score will be 70%.

Whenever the TOTP strategy is set up at a location, it supersedes the scavenging one, as it provides stronger guarantees. This way, updating the list of networks is not a concern for locations where custom APs are installed.

Because the process of announcing a network is independent of the number of nearby devices, this approach allows a virtually unlimited number of nearby devices to create proofs of their location without overloading the Wi-Fi network.

Validating the authenticity of Wi-Fi and Bluetooth devices is hard because the hardware identifiers can be spoofed; in fact, some systems do this as an intended privacy measure [Gooc]. Because this solution does not involve bi-directional communication with other devices or networks, as in many witness-based proof strategies [ZC11], it minimizes user exposure to attacks. This also protects their privacy, as only the entity operating the CROSS server will be able to know the locations visited by the user.

4.3.4 Kiosk strategy

The TOTP strategy prevents the attacker from creating new proofs on the fly, but not from replaying proofs from a legitimate visit under a different user account, or tunneling the information to a distant user. The kiosk strategy counters the possibility of claiming multiple rewards for a single trip, by preventing variants of Sybil attacks [DR02], where a malicious visitor creates multiple user accounts and runs them in parallel using one or more smartphones.[5]

This strategy requires the tourist to interact with a machine present at the location - the *kiosk* - in order to prove his presence. In CROSS, the main function of the kiosk is to sign a message for the CROSS client, logged on the account of the user and running on his smartphone. The kiosk can have other functionality unrelated to CROSS, including showing advertising or information about the location. Existing tourism information kiosks can be adapted for this purpose.

[5]When using just one device, the attacker would spoof its MAC address and other identifiers to pose as multiple devices, each with their own user account.

Proof production and validation

Similarly to Wi-Fi APs in the TOTP strategy, kiosks are required to have their clocks synchronized with the server, also with minute granularity. Each kiosk keeps a private key, used to sign information. The server has the corresponding public key. Kiosks do not need to have a network connection to the server.

Location proofs are produced as follows. The client application sends the username of the logged in user to the kiosk, by displaying a QR code that is scanned by the kiosk. The latter, using its private key, signs a message containing the kiosk ID, the username of the user, the current date and time, and a randomly generated large number (a nonce). This message and respective signature is sent back to the client, again using a QR code, which is scanned by the smartphone built-in camera.

The smartphone stores this data as a visit proof. When the trip log is submitted to the server, it verifies this proof by checking the signed message using the public key associated with the kiosk at the visit location; the username matches the user account submitting the proof; the date and time is contained within the period of the visit; the nonce was not reused from any other visit proof submitted in the past. The signed timestamp allows for limiting the validity of the proofs, eliminating the need to store nonces for a long period.

Unlike the scavenging and TOTP strategies, this one is binary: it either undoubtedly proves the presence of the user at the location, or provides no trace of it. Like the scavenging one, this strategy is also incapable of proving the duration of the visit, unless the kiosk device is more akin to an access gate and can monitor when the user enters or leaves the venue. The naive approach to mapping this strategy to a *confidence score* would be to assign two possible values, 0 for no visit and 100 for a proven visit. However, it is more interesting to use this strategy in conjunction with one of the previous two. The confidence score could then be that of the Wi-Fi based strategy, with a bonus multiplier if the kiosk validation is successful, or a penalizing multiplier if it is not.

By eliminating the remote network connection to the kiosk, an attacker must be physically present at the location to interact with it using QR codes. This physical interaction is essential to prevent Sybil attacks [DR02]. It can easily be inspected by a bystander, e.g. a tourist attraction staff member, who can check the behavior of the users for any suspicious activity, e.g. attempting to check-in with more than one device, or using the same device to check in multiple times, using different user accounts. An additional security mechanism could be the collection of ad-hoc witness reports from users in the same location, as described in [FP18]. However, this will not be necessary for this use case because kiosk devices can act as trusted witnesses.

The inconvenience for the user, and the kiosk setup cost for the system operators, can be greatly minimized by limiting the use of this strategy to a few locations per trip, where there are already

tourist support infrastructures.

4.3.5 Other considered techniques

We considered using Global Navigation Satellite Systems (GNSS), as explored in Section 2.1.1, to identify the location of the user and to enrich the produced location proofs. GNSS is the most common source of location information in Android applications, but it is also the easiest to manipulate. Both emulators and real devices support falsifying the location information returned by GNSS API functions, to help with development and debugging. This typically requires enabling the "Allow mock locations" setting, which can be detected by applications, but the incentive to cheat in location-based games, like those mentioned in Section 2.4.2, popularized more sophisticate ways to achieve the same effect, bypassing this setting.

GNSS also present reduced accuracy indoors, or may be outright nonfunctional. Finally, their use would almost certainly increase the power consumption of the device while registering visits. Because we only want to know whether the user was present at a location, not the precise coordinates of their path, we consider GNSS would not significantly improve the tamper resistance of our strategies. An attacker with the means to provide fake Wi-Fi scan results, most likely can easily tamper the GNSS location information provided to the application.

One possible alternative to the TOTP and Kiosk strategies is to have the client connect to an access point at the location, in order to establish a bidirectional communication channel. A challenge-response protocol, based on asymmetric cryptography, would be used to produce a location proof. This protocol would be conceptually similar to the one used in the kiosk strategy, with communication being carried out through Wi-Fi or Bluetooth instead of QR code scanning.

We decided against the implementation of such a strategy because, as explained previously, it would require connecting to possibly malicious wireless networks or devices, as verifying their authenticity before connecting is complex. It could also interrupt any ongoing Wi-Fi connection the device may have. Such strategy would not prevent Sybil attacks [DR02] where a single device presents itself as multiple ones by spoofing the hardware identifiers.

Another alternative to the TOTP strategy would be to use Wi-Fi Access Points or Bluetooth scanning devices to inform the server of which users are present at their respective locations. In addition to requiring a connection between the Access Points and the server, while still not protecting against Sybil attacks, Access Points are generally unable to know which clients are in range unless they are connected, which is to be avoided due to the aforementioned security implications. It is possible for some Access Points to enter promiscuous modes where the MAC addresses of nearby devices can be collected, but modern smartphones protect against this, by e.g. randomizing their

MAC address when scanning for networks [Gooc][MMD+17].

If Bluetooth scanning was used, the devices of tourists would need to be in discoverable mode, in order for their presence to be registered. This presents privacy issues, as anyone would be able to register this information for other purposes, and may pose energy consumption issues, as the mobile devices would need to actively broadcast their presence at regular intervals.

These strategies, like Scavenging and TOTP, are vulnerable to wormhole[6], signal amplification[7], and Sybil attacks. Their implementation would be more complex and prone to failure, have most of the disadvantages of the scavenging and TOTP strategies, while still not providing the same guarantees as the Kiosk strategy.

Wormhole and signal amplification attacks could be countered using distance-bounding protocols, as introduced in [BC]. These are unsuitable for our use case, as Android, like other smartphone operating systems, is not real-time and does not provide low-level hardware access, making it impossible to distinguish between random scheduler delays and those caused by such attacks.

The use of ambient-sound systems akin to those described in Section 2.1.3 was considered, to allow for a different kind of environment fingerprinting, but their use was ultimately decided against:

- Naturally, the use of these systems requires access to the microphone of the smartphone. On many mobile operating systems, this rightfully warns users that an application is using the microphone. Users may fail to realize why a "smart tourism application" would use the microphone in the background, augmenting privacy concerns even if all analysis is performed on the client;

- Audio analysis for prolonged periods of time will decrease the battery running time, especially if the radios and GNSS receivers are already being used to prove the location of the user;

- For many locations, sound fingerprints can vary heavily depending on the time of day and day of week, weather, or a number of other uncontrollable factors, especially outside;

- With many smartphones not leaving the pocket or purse of tourists for extended periods of time, obtaining audio input of sufficient quality could prove troublesome (at a limit, all fingerprints would be of the sound of phones moving around in pockets);

- Depending on the device and its specific configurations, access to the microphone can be impossible during phone calls.

[6] Attacks where packets are tunneled from one physical place to another.
[7] Attacks where a wireless signal is amplified to make its source appear closer than it actually is.

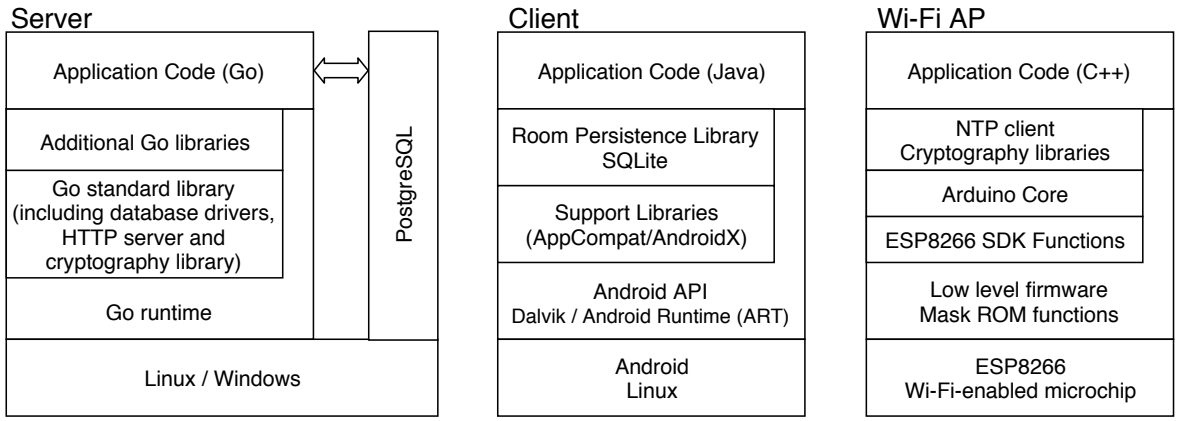


Figure 4.8: Overview of the used technology stack.

4.4 Technology stack

Functional server, client and customized Wi-Fi AP prototypes were developed, supported on different sets of technologies. These were used to evaluate our solution, as will be presented in Chapter 5. Figure 4.8 shows how the technologies discussed in this section are used in our solution.

4.4.1 Client application

The client application targets the Android operating system. We developed the client for this platform because access to peripherals, like sensors and wireless radios, is traditionally less hampered on Android, when comparing with other relevant smartphone operating systems. In the context of this work, this is important because the location proof techniques discussed in Section 4.3 require this less restricted access to the capabilities of these components, namely, the Wi-Fi scan results.

Wireless radio access is very restricted in iOS. As a relevant example, it does not provide any public, officially-supported API for Wi-Fi network scanning. iOS development is only officially supported on macOS and is primarily done using Swift, a language almost exclusively used in development for these two platforms. These factors led to the exclusion of iOS as a possible target for our prototype.

Development of Windows 10 Mobile has ceased and the market share of this operating system is negligible, excluding it as a possible target for our solution. It is also unclear whether this operating system ever allowed access to Wi-Fi scanning, or if it shares the limitations of iOS.

It should be noted that recent Android versions, namely the major versions 8 ("Oreo", API levels 26 and 27) and especially 9 ("Pie", API level 28) heavily restrict application access to some hardware capabilities, notably some that are useful for implementing location techniques, including the ability to scan for Wi-Fi access points.[8] Only starting with Android 10 ("Q", API level 29), an option was added, in the normally hidden developer settings, that disables this restriction. Because of this, our

[8] `https://developer.android.com/guide/topics/connectivity/wifi-scan#wifi-scan-restrictions`

solution targets version 8 ("Oreo"), where such restrictions can be worked around by keeping the application in the foreground, which can be achieved, for example, by displaying a notification.

Applications run on a Java Virtual Machine, and are able to use APIs provided by Android, as well as libraries included with the application code. The latter include Support Libraries, which provide features from recent Android versions even when the application is running on older ones.

SQLite is used in conjunction with the Room Persistence Library[9] in order to manage client-side storage (*Internal storage* and *Cache* in Figure 4.3).

The application was developed using the *Android Studio* IDE[10] and the *Gradle* build and dependency management solution, which corresponds to the currently officially supported Android development tool set.

4.4.2 Server

The server handles the HTTP API requests and queries the relational database containing the catalog and user information, in order to perform its core function, which is to verify location proofs, assign rewards and serve the catalog and relevant user information to each client.

The server is written in the Go programming language[11], a fairly recent language (released in 2009) but which has gained popularity in recent years. Go provides a very complete standard library, including cryptography packages implementing different encryption and hash algorithms, secure random number generators, and certificate manipulation functions. Go also provides libraries for connection with SQL databases, creation of HTTP servers and handling of requests, and serialization libraries, including a JSON decoder and encoder. These features make Go a good choice, as they facilitate HTTP server programming, database interaction, and location proof validation.

Go programs compile to a single statically-linked binary, which helps with their deployment. Go code is generally platform-agnostic and can be easily cross-compiled[12]. Because the resulting binaries are statically linked, their behavior is less prone to change depending on the configuration of the operating system where they are deployed.

If care is taken with regards to dependency management (namely, through the use of a practice known in the Go community as *vendoring*), Go builds are almost fully reproducible. We used *dep*[13] as the dependency management tool.

Another reason for the choice of Go is that we have validated its use, in solutions architecturally

[9] A library that abstracts the use of SQLite. `https://developer.android.com/topic/libraries/architecture/room`

[10] Integrated Development Environment

[11] `https://golang.org/`

[12] Compiled on an architecture other than the target of the resulting binary, e.g. producing a AArch64 Linux binary on x86-64 Windows.

[13] `https://golang.github.io/dep/`

similar to the one we are proposing, in past projects. We already had high proficiency in the use of a number of relevant libraries for database querying and transaction management, organization of REST handlers, serialization, authentication and cryptography.

The server uses a relational database with SQL support to store the catalog and user information. PostgreSQL is used as the database management system. Alternatively, MySQL or MariaDB could have been used. The interoperability of these three database management systems with the SQL database driver in Go had already been validated on previous projects. We decided to use PostgreSQL because we had higher familiarity with it.

In our prototype, configuring the server must be done directly using SQL statements, either through a command line client, or through a graphical tool such as pgAdmin. Configuring the server mainly consists on registering records of the different entities described in Figure 4.4 of Section 4.2. It would be essential for a production-quality product to include a web dashboard to configure and monitor the system.

4.4.3 Customized Wi-Fi Access Point

We developed a functional prototype of the customized Wi-Fi Access Point (AP) used in the TOTP strategy, described in Section 4.3.3. Such an AP could have been implemented using, for example, conventional Wi-Fi adapters connected to desktop and laptop computers, operating in AP mode.

We opted for a more realistic and portable solution, developed to run on programmable low cost embedded computers with Wi-Fi support: the ESP8266 microchips developed by Espressif Systems. These chips include a low-power Tensilica 32-bit CPU, that runs a proprietary RISC instruction set, at a default clock speed of 80 MHz and a maximum speed of 160 MHz. They include about 80 kB of static RAM (SRAM), of which about 50 kB are available for user applications, with the rest being used by basic hardware drivers and firmware support functions. The specific memory allocations vary depending on the operation mode, selected SDK[14] version, among other factors. The supported Wi-Fi protocols are 802.11 b, g and n on the 2.4 GHz frequency range [Esp19].

These chips, available packaged in inexpensive modules that include Flash memory, are programmed using C and C++, or higher-level languages including MicroPython[15] and Lua. Our prototype is programmed in C++ using the Arduino environment for this platform.[16] To aid with the computation of the TOTP to broadcast in the SSID, the Arduino Cryptography Library by Rhys Weatherley[17] is used.

[14]Software Development Kit, based on GCC, developed by Espressif for the ESP8266. The Arduino framework is implemented on top of this SDK.

[15]`https://micropython.org/`

[16]`https://github.com/esp8266/Arduino`

[17]`https://rweather.github.io/arduinolibs/crypto.html`

To implement the TOTP strategy, the AP must have a synchronized clock. Because most ESP8266 modules do not include a battery-powered real-time clock, the system must obtain the time when it boots, which can be challenging.

When the system is in range of a trusted Wi-Fi network with Internet access, this problem is solved quite easily. Because the ESP8266 can act as a Wi-Fi station and AP simultaneously, it can connect to the Wi-Fi network to periodically synchronize the volatile real-time clock included in the chip, over NTP[18]. For this purpose, the NTPClient library[19] is used.

When a trusted network is not available, because this is just a prototype, our implementation can also "scavenge" existing open Wi-Fi networks to obtain the current time and date. It will attempt to connect to each open Wi-Fi network in range, until it synchronizes the clock. In addition to NTP, it can also obtain the time and date from the Date header in response to a HTTP request. We found that this enables the system to obtain the time and date even when connected to networks with captive portals, such as those operated by or in partnership with Fon Wireless Ltd.[20], making this workaround successful in many urban environments.

It is very important to note that this workaround is completely unsuitable for use in production systems, as it is trivial for an attacker to provide their own open Wi-Fi network that will feed our AP incorrect time information, breaking the TOTP strategy – a denial of service attack, as no proofs produced using an AP with a wrong clock will be accepted by the server. In a production system, if a trusted Wi-Fi connection is not available, an alternative clock synchronization method would be needed, using, for example, a GPS receiver.

4.5 Usage example

Consider the following example scenario, where all components of our solution would be exercised, going over all steps necessary to set up our solution and use it in a production context.

Consider that Bob, who owns a tourism bus company, wants to give tourists an incentive to visit more places in town, hoping that this will lead to an increase in passengers transported by his company. Bob discovers CROSS and considers that the reward scheme it enables is an adequate solution.

4.5.1 Setting up the system

Bob, who in this situation will be the *system operator*, has already figured out that in order to increase the usage of his most profitable bus route, tourists should be incentivized to visit three

[18] Network Time Protocol

[19] `https://github.com/arduino-libraries/NTPClient`

[20] Company that operates the Fon hotspot network, focused on residential Wi-Fi sharing. `https://fon.com`

different attractions: a small public park (Park), a Monument and a Museum. The bus route goes through these attractions in this order, and Bob intends to reward tourists who complete this circuit with a discount voucher to the fancy restaurant operated by his cousin.

In the Park and at the Monument, the only viable location proof strategy is the Scavenging one, as the nearby residential and commercial buildings ensure there will be a decent amount of third-party Wi-Fi networks, but Bob does not have the permission to set up his own Access Points or a Kiosk. To set up the system for these locations, Bob will need to visit both and take note of the BSSIDs[21] of the Access Points in range at each location.[22] To easily exclude mobile and temporary APs, one solution is to visit each location in separate days and consider only those APs present in both visits.

Bob will then need to select which networks should be *triggers* in each location, i.e. the networks that the clients will know about, so they can detect their presence at the location and collect proofs. Based on experiments in the field, that we will present in Section 5.2.1, we recommend that about a quarter of the networks in range be used as trigger; preferably, these should be the networks with the strongest signal.

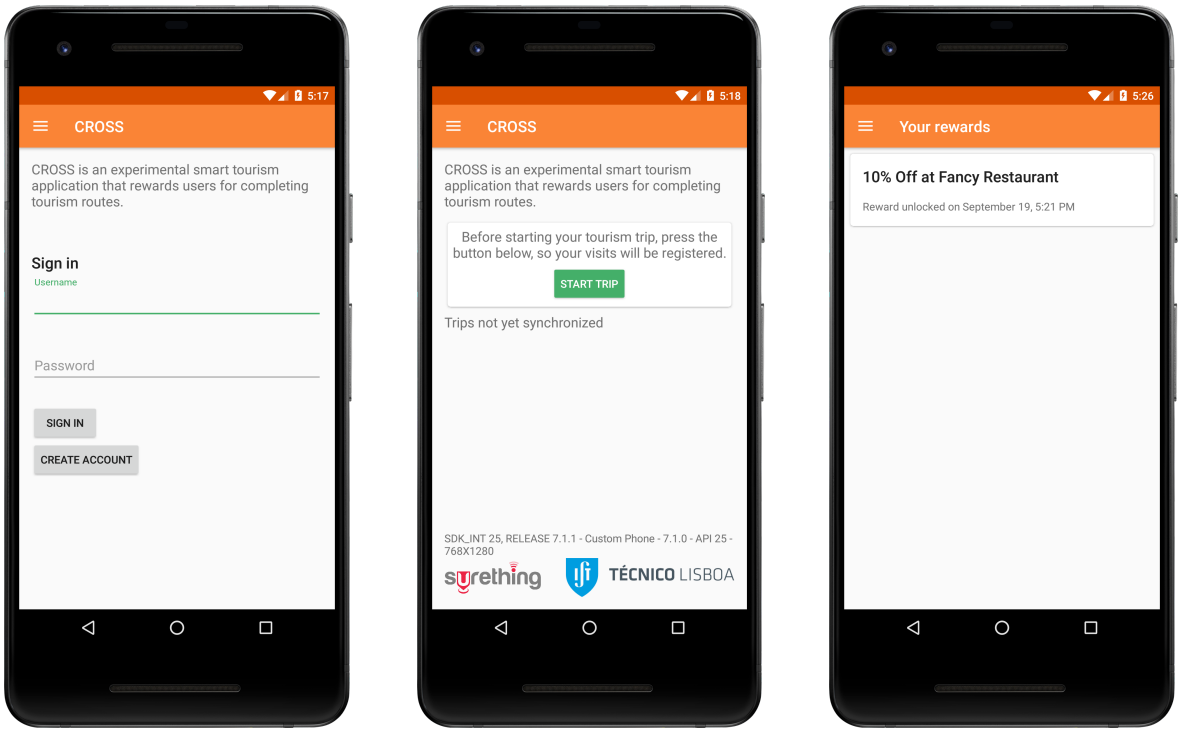
In the Museum, Wi-Fi coverage is lackluster. Bob signs a partnership with the Museum administration; together, they will install a network of APs providing free Internet to visitors and running the custom software necessary for the TOTP strategy. In this example, the Museum administrator is a *POI official*. In the TOTP strategy, all APs should be a trigger, because the secret elements are not the BSSIDs, but the SSIDs broadcasted by the APs.

Once all APs are registered in our solution, together with their TOTP secrets in the case of the Museum APs, Bob must configure the remaining entities: each location will be a *POI*, the *Route* will have the voucher *Reward* and three *Waypoints*, one per *POI*. Minimum visit durations must to be set for each *Waypoint*, as well as *confidence score* thresholds. Bob decides to set a minimum of 15 minutes for the Park and Monument, and a minimum of 30 minutes for the Museum. In the case of the Scavenging strategy, the ideal values for the thresholds can be hard to determine, as explained in Section 4.3.2. After testing the route, Bob considers that the system is finally ready to be used by tourists.

4.5.2 Using the system

Alice, a *tourist* looking forward to use the tourism buses, installs the CROSS client application on her smartphone. She will be prompted to create an account and sign in, as depicted in Figure 4.9.(a). She will also be prompted to grant the Location permission to the application. After signing in, Alice can go offline, which is convenient as she has not purchased a data plan. To preserve her

[21] Basic Service Set Identifiers, which in the case of an Wi-Fi Access Point, corresponds to its MAC address.
[22] This can be done using one of the many "Wi-Fi Scanner" or "Wi-Fi Analyzer" Android applications.



(a) Application sign-in screen.

(b) Main application screen after signing in.

(c) Screen showing the list of unlocked rewards.

Figure 4.9: Screens of the CROSS client application.

privacy, the application only collects data in visit logging mode, launched by pressing "Start trip", in Figure 4.9.(b).

Alice then visits the Park, the Monument and the Museum, in order. She spends 20 minutes in each of the first two locations, and 40 minutes at the Museum. During all this time, because the Kiosk strategy is not being used, Alice does not need to use her smartphone at all. If the Kiosk strategy was being used, for example, at the Museum, she would need to take a moment during her visit to exchange QR codes with the Kiosk, or she would otherwise not be eligible for the reward voucher.

Finally, Alice presses "End trip". Because she is offline, she won't immediately receive the reward. Later that day, when she connects to the hotel Wi-Fi, the application will submit the trip log and associated visit proof elements to the server. Because all visits were successfully detected and verified, she will receive the discount voucher for the fancy restaurant, as depicted in Figure 4.9.(c).

In our solution, reward attribution is a task to be solved by the *system operators*. Our system only determines whether the user should receive the reward, not how to securely do it. In this example, simply showing the rewards screen of the application to the restaurant waiting staff could suffice, but such a solution is so easy to fool that the whole effort becomes pointless. A better solution would be to integrate our system with the payment system at the restaurant.

4.6 Summary

We described the implementation of CROSS, a solution that, once configured by *system operators* and *POI officials*, lets them verify whether *tourists* using the system have verifiably visited predetermined locations. In Section 4.1, we described the use cases to be supported for each of these entities, as well as the desirable properties of the solution. As detailed in Section 4.2, our solution uses a client-server architecture, where communication occurs through a HTTP REST API. Location proof elements are collected by an application running on the smartphones of the tourists, and are validated by the server, which ultimately decides whether users have completed tourism circuits, unlocking rewards accordingly.

Visits are verified using three different location proof strategies with increasing tamper-resistance, two of which are based on Wi-Fi (Sections 4.3.2 and 4.3.3) and a third one based on physical interaction with kiosk-like devices (Section 4.3.4). Other considered location proof techniques and ideas were discussed in Section 4.3.5.

Client and server prototypes were developed, as well as a prototype of the customized Wi-Fi Access Point used in one of the location proof strategies. The client prototype is an Android application developed in the Java programming language. The server was programmed in Go and uses a PostgreSQL database. The Wi-Fi Access Point prototype was developed for the ESP8266 microchip, programmed in C++. These technology choices were discussed in Section 4.4.

Finally, a thorough example of how the prototype is meant to be set up and used by *system operators* and *tourists* was presented in Section 4.5.

Chapter 5

Evaluation

This Chapter encompasses our evaluation of the proposed system, CROSS. In Section 5.1, we will analyze the security of the system against different types of attackers. In Section 5.2, we will present the results of the evaluation of the prototype in a realistic setting, with a focus on the Scavenging and TOTP location proof strategies. Section 5.3 will present our power consumption analysis. Section 5.4 will discuss the results of the user survey we conducted. Lastly, in Section 5.5, we will analyze the feasibility of the Scavenging strategy based on data collected at various locations in the city of Lisbon.

5.1 Security assessment

The purpose of this assessment is to clarify what types of attacks CROSS is designed to counter and the limitations of the proposed solution in this regard. We consider an attacker model where the attacker has one or more of the following goals:

- *Tamper*: Obtaining more rewards than he has the right to, considering the routes he actually completed;
- *DoS*: Disrupting the CROSS system (denial of service), e.g., preventing other users from receiving rewards;
- *Hijack*: Attacking CROSS users through the CROSS infrastructure, by e.g. using it to spread malware.

Tamper is the most relevant goal, because the main contribution of CROSS is in the field of location proof systems. However, two concerns with existing witness-based solutions, and which led us to the selected strategies, is that they require cooperation between prover and witnesses, potentially compromising the Availability of the system, and the peer-to-peer connections could be

exploited for malicious purposes. Therefore, it is important to understand in what situations our solution is vulnerable to *DoS* and *Hijack*.

To model different types of attackers, we considered the capabilities presented in Table 5.1, divided in sets **A** through **D**. **A** focuses on server control, **B** affects the connection between clients and server, **C** consists on client control and client-side tampering, and **D** are generic infrastructure attacks.

A1	Read all the information in the server database.
A2	Take control over the server.
B1	Record all communication between the server and his client.
B2	Record all communication between the server and any client.
B3	Send requests to the server, posing as a client, using alternative software.
B4	Suppress communication between the server and any client, or redirect the client to a rogue server.
C1	Make the client application believe any Wi-Fi network is nearby.
C2	Know all the networks at a given location, at a certain point in the past.
C3	Know all the networks at a given location, in real-time, without being present.
C4	Set up rogue Wi-Fi Access Points.
C5	Know the OTP secret key of a AP used in the TOTP strategy.
C6	Pretend to be multiple users when at a kiosk.
C7	Know the private key of a kiosk.
C8	Set up fake kiosks.
D1	Send Wi-Fi deauthentication packets.
D2	Attack devices connected to the same network.

Table 5.1: Attacker capabilities considered in our security assessment.

Ideally, the solution should guarantee the CIA properties (Confidentiality, Integrity and Availability) regardless of the performed attack. In this context, Confidentiality corresponds to a lack of access to privileged information, which consists on the trip history of other users, network lists used by the Scavenging strategy, and OTP secret keys used by the TOTP strategy. Integrity corresponds to the inability to tamper with trip logs and data used in location proofs, including the forgery of location proof elements, and is directly tied to the *Tamper* attacker goal. Availability corresponds to the normal and continuous operation of the system, namely by continuing to let users download the *catalog*, create accounts, sign in, submit trip logs, and unlock the rewards they should qualify for. The latter property is directly tied to the *DoS* attacker.

Capability A1 can be acquired by discovering, for example, a vulnerability in the server REST API that lets the attacker perform arbitrary SQL read-only queries. A1 attackers are not able to change the data associated with other user accounts or impersonate users (passwords and API tokens are

hashed and salted) but they will immediately gain capabilities C2, C5, B4 and (partially) B2.[1] The system loses the confidentiality property against attackers with this capability. Because the server does not know the private key of each kiosk, it is still impossible for A1 attackers to break the third proof strategy. The integrity property is lost for the other two strategies. Availability is not affected.

A2 attackers essentially become as powerful as the system operators, and can manipulate the system to their own will, being able to achieve all three goals. In the CROSS security model, the server is the trust anchor, and therefore, it is not designed to counter attackers with capabilities A1 or A2. Attackers with capability A2 are not able to produce kiosk proofs, so while they achieve the *Tamper* goal, this tampering will be obvious if the proofs are independently audited. Confidentiality, Integrity and Availability are lost.

Capability B1 can be acquired by compromising the connection of a single client with a man-in-the-middle, and B2 by compromising the server connection. As long as B1 or B2 attackers are not able to obtain the clear-text content of the HTTPS connections between the server and the client, they will only be able to violate users' privacy by inferring usage patterns. They will not be able to achieve any of the three goals, and the system retains all the CIA properties.

Capability B3 can be obtained relatively easily, by reverse-engineering the client application to obtain the API address and request format. By itself, this capability does not let the attacker achieve any goals, nor does it affect CIA, but it is useful in conjunction with some of the C capabilities.

Capability B4 will let the attacker perform selective denial of service, as well as obtain the credentials of clients connecting to the rogue server, directly affecting the Availability and Confidentiality properties. To set up a rogue server, the attacker will need to obtain a valid TLS certificate for the FQDN[2] of the server that matches the pinned certificate in the client application. The most likely way to do this is to compromise the server and obtain the private key of the certificate, i.e. obtain capability A2, at which point setting up a rogue server is a pointless exercise.

C1 can be used in conjunction with C2, C3 and C5 in order to produce illegitimate visit proofs, breaking the Integrity property. With capability C2, an attacker can produce valid proofs for any location that uses the scavenging strategy, and can produce valid TOTP strategy proofs for the limited period in time when the list of APs was obtained. Therefore, the Integrity property is lost for these two strategies.

With capability C3, which can be acquired using signal amplification or by tunneling information from the attraction location to the attacker's location, an attacker can produce valid proofs for both the scavenging and TOTP strategies. Again, these two would lose the Integrity property. In

[1] Communication between server and client can be inferred from the database data, because some of the communication will result in changes to the data.

[2] Fully-qualified domain name

conjunction with capability B3 or C1, they will therefore be able to achieve the *Tamper* goal with routes that do not use the kiosk strategy.

An attacker with capability C4 can achieve the *DoS* goal by faking triggers, making client applications believe they are at a different location, breaking the scavenging strategy. They can also break the TOTP strategy by making clients collect SSIDs with nonsense OTPs, producing invalid proofs that will be rejected by the server. This capability does not help attackers achieve the *Hijack* goal, as clients never automatically connect to any of the CROSS Wi-Fi networks. Capability C4 will cause the Integrity and Availability properties to be lost for the TOTP and Scavenging strategies.

An attacker that acquires capability C5 will be able to fraudulently prove his presence at the location of the AP, at any past and future point in time, until a new key is generated for that AP. This allows him to achieve the *Tamper* goal, causing the TOTP strategy to lose the Integrity property, but only if the stronger kiosk strategy is not used in some other point of the route.

Capability C6 will let an attacker produce valid proofs for the strongest strategy. In conjunction with capability C4 and C1, this lets them achieve the *Tamper* goal with any route. The Integrity property is lost for the Kiosk strategy. We believe that acquiring this capability is complex, as long as users are monitored by a supervisor when operating the kiosks. An alternative capability that nets the same results is C7, which can only be acquired by tampering with the kiosk to extract its key, or compromising the key when it was configured in the kiosk.

With a naive implementation of the Kiosk strategy, where the client application does not verify the authenticity of the kiosk, an attacker with capability C8 would achieve the *DoS* goal by tricking clients into collecting invalid proofs which will then cause the rejection of the trip log. However, if a mutual authentication scheme is used – one can be easily implemented, by letting clients know the public key for each kiosk – the application will be able to detect a fake kiosk. The only way to set up a functional impostor kiosk is by also having C7.

Attackers with capability C8 may also achieve the *Hijack* goal, if they can exploit the QR reading functionality of the client application or of the kiosk. QR codes can be used as an attack vector, as summarized by Krombholz et al. [KFK+14]. We believe our system is not vulnerable to the attacks mentioned in their work. The QR code reader in CROSS does not act on any URL, *tel:* command, or similar contents. In CROSS, the purpose and context of the QR codes is clear for the users, rendering unfeasible attacks similar to the *QRishing* experiment.

Capabilities D1 and D2 do not affect any of the proof strategies, as they do not require the user to be connected to a network. This is one of the advantages of not using peer-to-peer communication between devices: because no connections are established, the potential for attacks and privacy invasion is much smaller. However, D1 and D2 can help the attacker achieve the *DoS* goal, if they

prevent other users from submitting their trip logs, therefore affecting the Availability of our system for these users.

Overall, we can conclude that the security mechanisms in place for the smart tourism application are well suited. The cost of attacks with the *Tamper* goal exceeds the value of the intended rewards, which are small (e.g. a value of up to EUR 2) and will likely consist on discount coupons whose full redemption will require a purchase. The cost of *DoS* is high and there is no clear benefit for a specific attacker. The benefit of *Hijack* for an attacker is limited, because most times there will be no direct device interaction – the device sees the networks but does not connect to them – or is limited to the scanning of a data QR code.

5.2 Field experiments

An evaluation scenario was set up in the Alameda campus of Instituto Superior Técnico. Three locations were selected to simulate a tourism route, shown in Figure 5.1 as locations **A** through **C**. Additionally, a control location, **N**, not part of the route, was selected. Participants were asked not to visit this location. The characteristics of each location are shown in Table 5.2.

Location	Common Name	Type	Area
A	"Jardim Sul"	Outdoor garden	$\approx 3300\,m^2$
B	"Jardim Norte"	Outdoor garden	$\approx 3900\,m^2$
C	"Room 0.09 – Pavilhão de Informática III"	Indoor room	$\approx 31\,m^2$
N	"Átrio do Pavilhão Central"	Indoor hall	$\approx 526\,m^2$

Table 5.2: Characteristics of each location considered in the experiment.

The route was configured in the system such that the three locations had to be visited in order, from **A** to **C**. The Scavenging location proof strategy was used in locations **A**, **B** and **N**; the TOTP strategy was used in location **C**. Details about the route configuration are shown in Table 5.3. "Minimum visit duration" corresponds to the minimum acceptable duration configured in the system; "Effective visit duration" corresponds to the amount of time participants were asked to stay at each location.

Location	Location Proof Strategy	Known APs	Trigger APs	Minimum visit duration	Effective visit duration
A	Scavenging	21	4	10 seconds	3 minutes
B	Scavenging	17	8	1 minute	3 minutes
C	TOTP	1	1	8 minutes	10 minutes
N	Scavenging	10	2	-	No visit

Table 5.3: Configuration of the experimental locations in our solution.

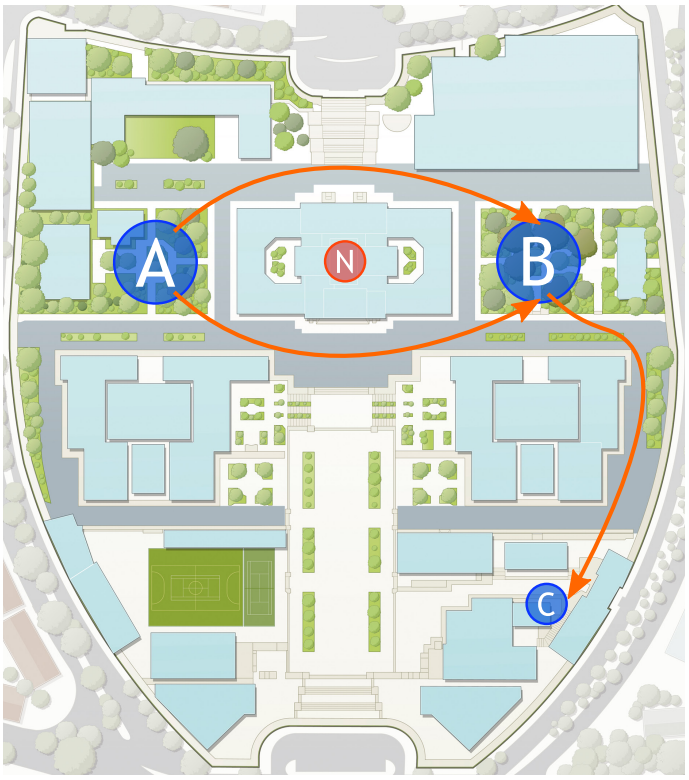


Figure 5.1: Alameda campus route used in the evaluation.

Figure 5.2: Representation of each Android version in the sample of devices used for evaluation.

The lists of APs for locations **A**, **B** and **N** were obtained immediately before the start of the tests, using the "Wifi Analyzer" Android application.[3] The networks with strongest signal levels at each location were selected to be triggers. At location **C**, only our customized AP prototype was used.

We conducted tests with voluntary participants who brought their own personal Android phones. This let us reach a large and diverse sample size, and approximates the experiment to the intended real-world use of the system, where there is no control over what devices tourists will use. A total of 34 Android smartphones were used in the experiment. A break down of the OS versions of these devices is shown in Figure 5.2.

Participants were told to assemble at location **A**. To make the experience more amenable, the CROSS application was published to the Google Play store, for participants to download. We hosted the server prototype on an Internet-facing machine, which the public version of the client application

[3] `https://play.google.com/store/apps/details?id=com.farproc.wifi.analyzer`

was configured to use.

Users were instructed to turn on the Wi-Fi and Location options on their smartphones, and to grant the location access permission to the application. Regarding the use of the application, users were asked to create a user account, to start the trip logging on command, and to only stop it when instructed, at location **C**, where participants answered the survey discussed in Section 5.4. Participants were told to put their phones in standby mode and carry them as they normally would.

It should be noted that our system was designed with longer visit durations in mind, with a minimum of ten minutes per location. Visit length was shortened for the experiment, to decrease the burden on the voluntaries. The application was configured to scan for networks every 30 seconds. Longer visit durations would ensure that the smartphones would see a larger set of networks, including those with weaker signals, as not all networks are immediately found on the initial scans.

To make the experience practical for all participants, all locations are inside the Alameda campus. Unfortunately, due to their proximity, the protection against travel time inconsistencies, described in Section 4.3.1, was not exercised.

Two sessions of tests were conducted, on separate days, in order to maximize the number of participants. In each session, users were divided in small groups, in order to better guide them through the testing procedure. In order to facilitate the statistical analysis, we made sure that all groups from both sessions followed the same procedure, and confirmed that other relevant environment variables, such as nearby Wi-Fi networks, did not change. Therefore, we do not feel a separate analysis per group or per session is necessary or adequate.

We decided to focus this part of the evaluation on the following aspects:

- Location detection performance: how often are locations correctly detected? Are locations detected within a useful time frame?
- Location proof performance: what are the optimal threshold settings that maximize the number of accepted visits while rejecting fake proofs?

5.2.1 Location detection performance

When evaluating the location detection capabilities of CROSS, we were interested in understanding whether locations are detected by the client application. We are not concerned about whether sufficient location proofs were collected, or even if the trip logs were successfully sent to the server.

In this experiment, the expected result is that each device should be able to detect locations **A**, **B** and **C**, and not detect location **N**. The results presented in Table 5.4 correspond to the results after the devices were present for three minutes at each location, except for location **N**, near which every device passed on the way between **A** and **B**.

Location	Total visits	Total detections	Success rate
A	34	30	88%
B	34	33	97%
C	34	34	100%
N	0	0	100%

Table 5.4: Location detection performance after three minutes at each location (except for **N**, not visited).

Discussion

As expected, no devices detected control location **N**. For other locations, results are satisfactory as well. The lower detection rate of location **A** in comparison with **B** may be explained by the lower number of trigger networks configured for **A**. As shown in Table 5.3, about one fifth of the networks in **A** are triggers, while in **B** about half of the networks are triggers. These are the results backing our affirmation, in Section 4.5.1, that at least a quarter of the APs known at a given location should be set as triggers.

All devices detected location **C** within three minutes, which may be explained by the fact that the single AP was in the same room as the participants, therefore its signal was much stronger and easier to detect than the signals of the APs at **A** and **B**, which were installed in the nearby buildings, at distances between 20 and 80 meters from the users.

It is also interesting to note that although almost half of the devices in the sample were running Android 9.0, the limitations imposed in this version and mentioned in Section 4.4.1 did not seem to affect the location detection performance of these devices. This is most likely because the scanning period of 30 seconds does not exceed the limit of four scans per two minutes imposed by this Android version.

5.2.2 Location proof performance

Because, in CROSS, location proof elements are analyzed by the server, in this section we will only consider visits that were submitted to the server as part of trip logs. For two of the participants, the trip log could not be submitted due to a bug in the client application, that let these users proceed to the main screen despite the account creation having failed. In addition, as we showed in the previous section, a minority of devices failed to detect some locations. The total number of trips analyzed per location in this section of the evaluation is shown in Table 5.5.

In locations **A** and **B**, the Scavenging strategy was used. In this strategy, the confidence score corresponds to the percentage of networks found by the client, compared to the total number of APs registered in the server for each location.

Location	Total visits	Total submissions
A	34	28
B	34	31
C	34	32

Table 5.5: Visits submitted to the server for analysis, per location.

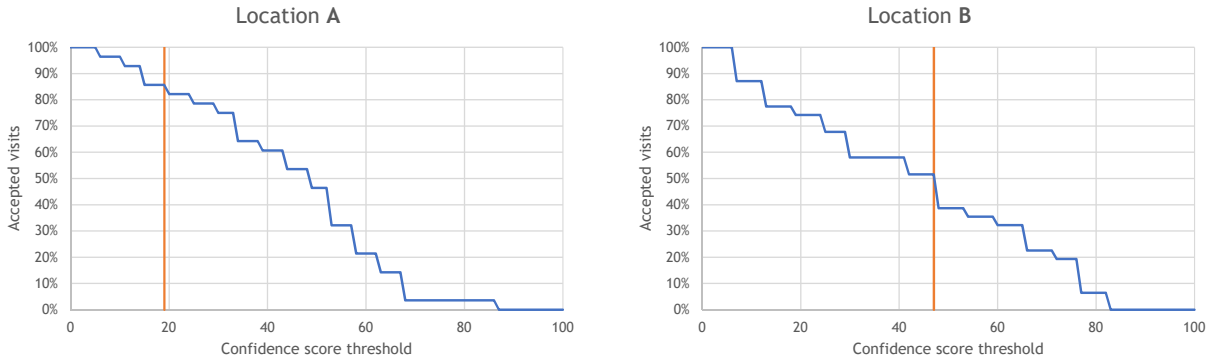


Figure 5.3: Percentage of accepted visits in function of the confidence score threshold configured at locations **A** and **B**.

Figure 5.3 shows the percentage of accepted visits for locations **A** and **B**, in function of the confidence score threshold that is set for those locations. When deciding whether to reward an user, all visits must be accepted for the trip to count, but here, each location is being analyzed individually. The vertical orange line in the charts corresponds to the percentage of known networks that are triggers, at each location. We consider that it represents the minimum confidence score threshold acceptable, as only visits proofs with a higher score are guaranteed to contain a non-trigger (secret) network.

In location **C**, the TOTP strategy was used. In this strategy, the confidence score corresponds to the percentage of visit time that could be verified by the TOTP codes present in the scan results collected by the client. Figure 5.4 shows the relation between the threshold and the accepted visits, for this location. Again, the visit is being analyzed individually, and the acceptance rate is for the visits to the location, not the whole trip.

Discussion

Results for the Scavenging strategy (locations **A** and **B**) fell short of expectations, as the confidence score threshold has to be set very low – lower than recommended – for a large percentage of visits to be accepted. For location **A**, the minimum recommended threshold would be 19, but that would only accept 24 of the 28 visits (86%), rejecting four legitimate visits. For location **B**, the minimum recommended confidence score threshold would be 47, accepting only 16 of the 31 visits (52%).

These results show that most devices did not see a majority of the networks associated to each

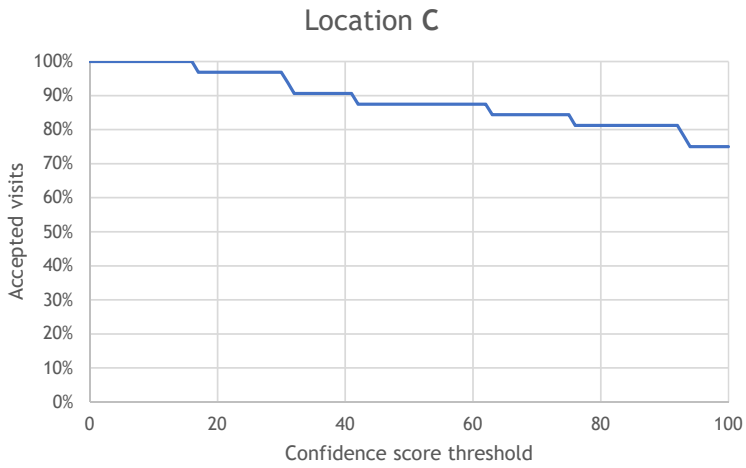


Figure 5.4: Percentage of accepted visits in function of the confidence score threshold configured at location **C**.

location, in part certainly due to the short visit duration (three minutes) and the weak network signal levels, whose APs were relatively distant.

Defining a larger ratio of trigger networks per location increases the probability that the location will be detected successfully, but because it increases the minimum threshold, it also means more visits will be rejected. The problem would be minimized if the strength score for this strategy was calculated based solely on the amount of non-trigger (secret) networks submitted by the client. However, the cause of the phenomenon would still exist: in the Scavenging strategy, the more networks that are set as trigger, the less secret information there is to consider.

Results for the TOTP strategy (location **C**) were positive. We believed most devices would have trouble capturing the SSID changes every two minutes, especially given the Wi-Fi scanning limitations presented in Section 4.4.1. However, this did not appear to be the case: 24 devices (75%) were even able to capture TOTP codes attesting the entirety of the visit period (10 minutes).

For the TOTP strategy, we do not recommend a minimum confidence score threshold; it is entirely up to the *system operators* to be more or less demanding. The results show it is possible to require that half of the duration of the visits be backed by TOTPs, and still accept over 80% of the visits. The positive results may be explained by the close proximity to the AP, which was in the same room as the participants.

Overall, the difference in results shows that the confidence scores for each strategy and even for each location are not directly comparable or mixable. In its current form, the system requires careful, manual and empirical calibration for each location and tourism route one may want to support. One clear challenge for future work is the development of a method for identification of the ideal thresholds and trigger networks for each location, eventually even recognizing whether the Scavenging strategy is suitable for a location, or if APs and other hardware, like Kiosks, will need to be deployed

specifically to support the system.

5.3 Power consumption

To assess the power consumption of our techniques and compare their consumption with that of alternative solutions, we collected battery usage data on a LG V40 ThinQ smartphone, running Android 9.0. To minimize noise effects caused by variation of irrelevant external factors, all data was collected with Bluetooth and the cellular radios disabled. Wi-Fi was enabled, but not connected to a network - the device was kept offline. We also kept the battery level between 50% and 80%, as to avoid the fudging introduced by Android when the battery is full or almost empty.[4] The device had a minimal amount of applications installed beyond those that came pre-installed, and we ensured that the lowest possible number of applications was running in the background.

We compared three different situations: location using both Wi-Fi and GNSS, location using exclusively Wi-Fi scanning, and no location collection at all. For the first case, a modified CROSS application, that also used GNSS to collect location information, was used. In the second case, the unmodified CROSS application was used. In both cases, data was requested every 30 seconds. In the third case, no applications were used - the phone was left turned on, with Wi-Fi enabled, without explicitly using any applications. Table 5.6 presents the results.

Method	Polling rate	Total test duration	Average battery drain
CROSS using GNSS and Wi-Fi	30 s	8 h	1.25 p.p.[5] / hour
CROSS using Wi-Fi	30 s	39 h 30 min	0.61 p.p. / hour
No collection	N/A	29 h 5 min	0.58 p.p. / hour

Table 5.6: Battery drain depending on the location collection method.

It is clear that the use of GNSS for location consumes more battery than using Wi-Fi scanning exclusively, with the device using, on average, about twice as much power than when not collecting location data. CROSS, which exclusively uses Wi-Fi, presents a negligible increase in power consumption relative to no location collection. This increase, of 0.03 percentage points per hour, can be attributed to random variations and to deficiencies in the analysis method, as the battery level is only ever presented by Android with unit precision.

[4] Above 90%, occasional topping charges are performed despite the device always reporting 100%. When the battery is almost empty, the percentage may decrease more quickly to avoid battery depletion and damage.

[5] Percentage points

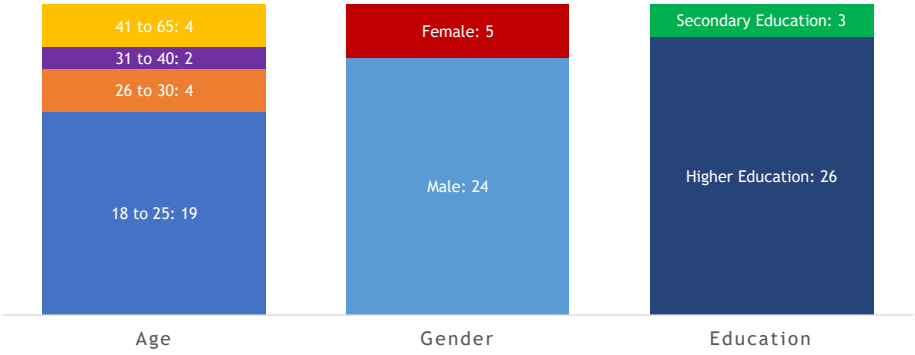


Figure 5.5: Characterization of the survey sample.

5.4 User survey

We conducted a small survey among the participants of the field experiments from Section 5.2. Users were asked to fill a multiple-choice questionnaire while they waited at location **C**. With this survey, we wanted to answer the following questions:

- How do users treat location information that is relative to them?
- How familiar are users with location proofs?
- What communication technologies, to produce and transmit proofs, are typically available on their devices?
- What are the most important criteria users consider when installing an application?

We received a total of 29 answers to the questionnaire. The characterization of the sample is shown in Figure 5.5. The sample is not representative of the general population: 24 of the respondents were male, 19 of them were less than 26 years old, and 26 attained higher education. We believe the results are still interesting, even if they only characterize a range of the population.

An overwhelming majority of the respondents (26, or 90%) does not commonly share their location on social networks. Generally, those surveyed were less willing to share their location with other parties, the less intimate those parties were, and the more connected to their identities the information would be. Only two respondents were unaware that smartphones can be constantly reading sensor data.

Notably, 20 of the 29 respondents were willing to share location information and history with a company or organization they trust, if such information was completely anonymous (not pseudonymized). On the contrary, only three were willing to share with the whole world if it was pseudonymized, and none of the surveyed were willing to share their location information with

Figure 5.6: Familiarity of the survey respondents with the concept of "location proofs".

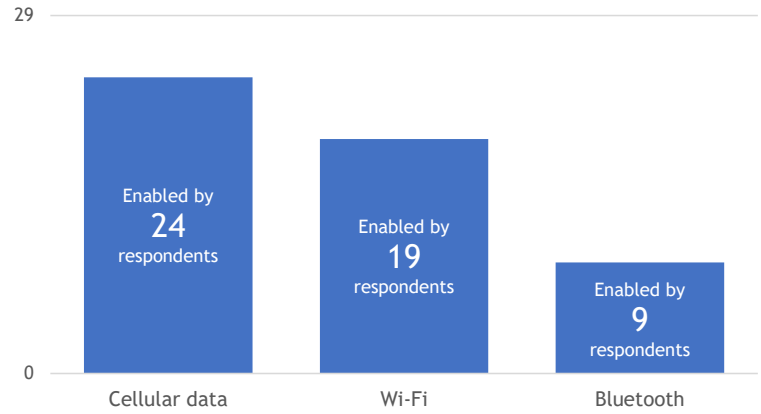


Figure 5.7: Connectivity options usually enabled by the surveyed.

the whole world if it had their real name associated. We can conclude that, for the population range represented by our sample, personal location information privacy is very important.

As shown in Figure 5.6, most surveyed volunteers were familiar with the concept of "location proofs". 90% of the respondents had an interest in being able to produce location proofs using their smartphone, for one purpose or another. 70% of them would be willing to collect location proofs for their own personal reference, while 67% would be willing to do so in order to justify their absence at work or school or to prove their presence at a public service.

We asked users whether they typically had mobile data, Wi-Fi and Bluetooth enabled on their smartphones. The intention is to understand what technologies can be used by location proof strategies. The results are shown in Figure 5.7. A majority of participants enables mobile data, but our sample is not representative of tourists, the target audience for our solution. It is possible that this figure would be lower among international tourists. The fact that a third of the surveyed does not usually have Wi-Fi enabled surprised us. Fortunately, many recent Android devices support scanning for networks even when Wi-Fi is disabled, thus, our location proof strategies will still work. These results validate our decision to base our strategies on Wi-Fi and not Bluetooth.

Respondents were asked to select the top three criteria they took into account when deciding whether to install an application. The most selected option was "Privacy and permissions", selected by 17 (59%) of the participants, again confirming the high privacy-consciousness of our sample. The second most selected option was "User interface appearance and ease of use". Complete results are shown in Figure 5.8.

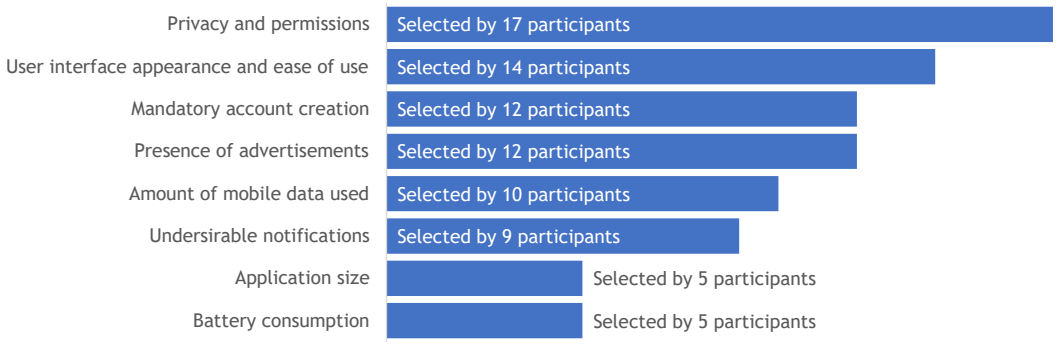


Figure 5.8: Most important criteria considered when installing an application, according to the respondents.

Overall, for CROSS to match the expectations of the users, we believe it should have an even stronger emphasis on privacy protection. It would also need to have a more polished user experience, compared to that of our prototype, whose focus was on the validation of the Scavenging and TOTP techniques. Given the high percentage of respondents who can use cellular data connections, it may be viable to implement different location proof strategies that communicate in real-time with the server.

5.5 Scavenging feasibility

One of the concerns with the scavenging strategy, presented in Section 4.3.2, is the need to maintain the lists of Wi-Fi networks for each location where this strategy is used. As time passes, some of the networks may disappear, and new, different networks may appear. Even though the server suggests the addition and removal of networks based on the submitted visit proofs, these suggestions need to be manually vetted.

If the sets of networks at a location change too frequently, the scavenging strategy may prove to be inadequate for that location: at a limit, the server database would need to be updated almost daily. The system would also need to be modified, to decide proof acceptance based not on the current set of known networks, but on the set that was in effect at the time when the proof was produced.[6] Therefore, it is important to understand how frequently Wi-Fi networks appear and disappear in the real world, to assess whether the current implementation is adequate.

We collected data on the Wi-Fi networks in range, at six locations in Lisbon, in three dates. The second date was ten days after the first, and the third date was 31 days after the first. Five of the locations are well-known tourist attractions and one is a residential area, that serves as an example of a location where there could be an interest in using the system, despite not being a recognized tourist attraction. Details of each location are presented in Table 5.7. We will mention locations by

[6] Recall that proofs can be produced offline and their submission is deferred until the device is online.

their *Reference name*.

Reference name	Name	Coordinates	Type
Alvalade	R. Eugénio de Castro	38°45'16.4"N 9°08'48.3"W	Residential area
Comércio	Arco da Rua Augusta	38°42'30.2"N 9°08'12.1"W	Landmark
Gulbenkian	Jardim da Fundação Calouste Gulbenkian	38°44'13.7"N 9°09'12.0"W	City park
Jerónimos	Mosteiro dos Jerónimos	38°41'50.9"N 9°12'21.5"W	Monastery
Oceanário	Jardins da Água	38°45'45.1"N 9°05'38.7"W	City park
Sé	Largo da Sé / Jardim Augusto Rosa	38°42'34.8"N 9°08'01.3"W	Church

Table 5.7: Locations where Wi-Fi networks were collected.

The application used to collect the information was a modified version of CROSS, which registered location information and Wi-Fi scan results every 30 seconds. Visits to locations lasted for 15 minutes each, and data was simultaneously collected by three different smartphones, in order to always collect the largest amount of networks possible and minimize random variations where a device may not see a network for unknown reasons.

We are interested in knowing, for each location, how many Wi-Fi networks are still *Present*, and how many *New* networks became available, after ten days and after a month. The results, presented in Table 5.8, correspond to the deduplicated network counts after merging the data from the three devices. Across devices and visits, APs were identified by their BSSID to avoid counting renamed networks (such as in our own TOTP strategy) as separate networks. Values for both periods are always relative to the first visit.

		After ten days		After one month	
Location	Initial total	Present	New	Present	New
Alvalade	86	74 (86%)	13	73 (85%)	31
Comércio	133	8 (6%)	60	7 (5%)	43
Gulbenkian	80	54 (68%)	92	54 (68%)	55
Jerónimos	148	34 (23%)	100	24 (16%)	62
Oceanário	39	22 (56%)	41	24 (62%)	40
Sé	61	25 (41%)	43	22 (36%)	44

Table 5.8: Wi-Fi networks present at each location after ten days and after a month.

5.5.1 Discussion

One positive aspect, seen in the results of Table 5.8, is the large amount of Wi-Fi networks present at each location, which is a prerequisite for the scavenging strategy. However, in certain locations, notably, Comércio and Jerónimos, most networks appear to be temporary. This can be confirmed by looking at their SSIDs: many clearly correspond to personal mobile Wi-Fi hotspots and smartphones, with names such as "iPhone", "VodafoneMobileWiFi", "NOS_Internet_Movel" or "Nokia 7 plus".

For the scavenging strategy we propose, temporary or mobile Wi-Fi networks should not be considered. The number of networks still present ten days after the first visit is a good indicator of the number of networks that can be considered in the scavenging technique, at each location. Most locations have a sufficiently large set of usable networks, with the notable exception of Comércio, where just 8 APs appear to be permanently installed.

In terms of the frequency at which the lists of networks must be updated, we can look at the number of permanent networks that disappeared between the second visit (after ten days) and the third visit (after one month). In most cases, there is only a minor reduction from one visit to another, with Jerónimos being the worst case, where 10, or 30%, of the permanent networks seem to have been disabled after twenty days. This is the only case where we believe frequent updates, e.g. bi-weekly, may be warranted, to ensure the correct operation of the system.

The location in a residential area, Alvalade, is the one that presents the highest number of permanent networks and also the highest ratio of permanent to temporary networks. Most of these networks seem to correspond to domestic routers of the nearby homes, and it makes sense that this set of networks would not see significant changes too frequently. The much lower number of tourists in this location explains why there are very few temporary networks, such as those from mobile hotspots.

The case of Oceanário is interesting, as two of the networks seen on the first visit have reappeared after one month, despite being absent on the second visit. These correspond to mobile Wi-Fi networks, for example in buses, that are only present sometimes, and which should not be considered in the scavenging strategy.

In the general case, the scavenging strategy appears to be viable and to not require excessive maintenance of the lists of networks on the server side. The suggestions given by the server on what networks to add and remove, can be used to update the lists without revisiting the locations, if one is willing to trust the trip logs of the users. At the very least, they can indicate that a new thorough survey of the networks is necessary. As mentioned in Section 5.2.2, the identification of which networks to consider, which networks to set as trigger, and what thresholds to set, continues to be the biggest challenge with this strategy.

In future work, more robust variations of the scavenging strategy could take advantage of the transient aspect of mobile Wi-Fi hotspots, in schemes similar to those used by witness-based location proof strategies. For example, when verifying proofs, instead of comparing exclusively against a fixed list of known networks, the server could also compare against other proof submissions from roughly the same time, and increase the confidence score of the proofs which contain the then present temporary networks.

With such a strategy, the system could become susceptible to a new type of collusion attack, where two or more attackers, or a single attacker using multiple identities, mention the same fake temporary networks in their forged proofs. This would boost their confidence score past the acceptance threshold, despite insufficient knowledge of the permanent networks present at a location. The means to reconcile proof acceptance and reward assignment after the initial submission would need to be developed, as confidence scores for past proofs could change at any moment, by later submissions confirming the presence of temporary networks.

Chapter 6

Conclusion

We presented CROSS, a system that implements location proof techniques in consumer mobile applications. Location proofs allow for the verification of the location information provided by smartphone sensors, increasing the dependability of this information. We used smart tourism as the demonstrative use case, developing a smartphone application where location proofs are used to implement a reward scheme. Users unlock rewards by verifiably completing predefined tourism circuits.

We proposed three different location proof strategies, two of which are supported on Wi-Fi and a third one which uses QR codes for human-visible communication with kiosk-like devices. Two of the techniques can verify the date and time of visits to locations, and therefore help establish location provenance. Our system is able to perform location detection using Wi-Fi exclusively, collecting information only when explicitly enabled by the user and only when in the vicinity of a registered location. By using multiple strategies, CROSS is easily adapted to the conditions of different locations and can provide varying degrees of tamper-resistance.

In CROSS, the Wi-Fi-based techniques do not require bidirectional communication between devices. In contrast with most witness-based techniques, CROSS does not use peer-to-peer communication. This allows the system to have the same performance independently of the number of users at each location. Additionally, we believe this has advantages in terms of user security and privacy, while providing a better user experience in current consumer mobile platforms, where peer-to-peer communication has become a secondary concern.

Often, compromises must be made between security, user experience and compatibility with existing technologies. We analyzed other location proof systems, some of which are more resilient against wider categories of attacks, and concluded that, for the smart tourism use case, a stronger emphasis in user experience and compatibility was necessary. We believe our solution provides well suited security mechanisms for this use case.

A prototype of CROSS was implemented, encompassing a Android application, a server application, and a custom Wi-Fi Access Point. The prototype includes the two location proof strategies based on Wi-Fi, which were evaluated regarding their accuracy, response times and configurations for optimal operation. Evaluation was conducted in a realistic setting, with 30 volunteers who used 34 different devices, and had positive results. We compared the power consumption of Wi-Fi-based location, used by our system, with GNSS-based location, and showed that CROSS uses comparatively negligible amounts of power. Finally, we collected and analyzed data regarding the Wi-Fi networks present near five tourist attractions, confirming that techniques which take advantage of existing Wi-Fi networks, are generally viable. Overall, we believe CROSS is a viable option for implementing location-based reward schemes, in the context of a mobile application.

In terms of future work relative to the proposed location proof strategies, the scavenging strategy would benefit from the implementation of a method for identification of the optimal threshold setting and trigger networks. Taking advantage of mobile hotspots for location proofing could also be valuable.

The CROSS prototype can be easily extended to support additional location proof strategies, however, the system would benefit from more robust means of combining them, instead of using a fixed hierarchy where certain strategies always completely supersede others. It would be extremely valuable to have the means to prevent Sybil attacks, without resorting to the proposed solution of visible interactions with a kiosk, which represent an interruption for the tourists.

Regarding user privacy, the proposed techniques are already privacy-protecting, by not broadcasting the location of the users, not disclosing their presence to others, and not collecting location information outside of predetermined locations. CROSS could benefit from further work in this area, by e.g. not disclosing the visit history to the system operators, or by making it impossible to associate a trip log with a specific user, while still assigning rewards.

Bibliography

[ACC09] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. SurroundSense. In *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09*. ACM Press, 2009. doi:10.1145/1614320.1614350.

[AHD+16] Ioannis Agadakos, Per Hallgren, Dimitrios Damopoulos, Andrei Sabelfeld, and Georgios Portokalidis. Location-enhanced authentication using the IoT. In *Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC '16*. ACM Press, 2016. doi:10.1145/2991079.2991090.

[App] Apple Inc. Core Location — Apple Developer Documentation. Accessed November 30, 2019. URL: https://developer.apple.com/documentation/corelocation.

[Bau13] Christine Bauer. On the (in-)accuracy of GPS measures of smartphones. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia - MoMM '13*. ACM Press, 2013. doi:10.1145/2536853.2536893.

[BC] Stefan Brands and David Chaum. Distance-bounding protocols. In *Advances in Cryptology — EUROCRYPT '93*, pages 344–359. Springer Berlin Heidelberg. doi:10.1007/3-540-48285-7_30.

[BDR07] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263, 2007. doi:10.1504/ijahuc.2007.014070.

[Ber10] Berg Insight. GPS and mobile handsets, 2010. Accessed November 30, 2019. URL: http://www.berginsight.com/ReportPDF/Summary/bi-gps4-sum.pdf.

[BJL13] Artur Baniukevic, Christian S. Jensen, and Hua Lu. Hybrid indoor positioning with Wi-Fi and Bluetooth: Architecture and performance. In *2013 IEEE 14th International Conference on Mobile Data Management*. IEEE, jun 2013. doi:10.1109/mdm.2013.30.

[Bra08] Daren C. Brabham. Crowdsourcing as a model for problem solving. *Convergence: The International Journal of Research into New Media Technologies*, 14(1):75–90, feb 2008. doi:10.1177/1354856507084420.

[BS03] A.R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, jan 2003. doi:10.1109/mprv.2003.1186725.

[Can08] Canalys. Canalys research release 2008/082, 2008. Accessed November 30, 2019. URL: https://www.canalys.com/static/press_release/2008/r2008082.pdf.

[CCCP13] Eyup S. Canlar, Mauro Conti, Bruno Crispo, and Roberto Di Pietro. CREPUSCOLO: A collusion resistant privacy preserving location verification system. In *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*. IEEE, oct 2013. doi:10.1109/crisis.2013.6766357.

[CY06] Scott Contini and Yiqun Lisa Yin. Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In *Advances in Cryptology – ASIACRYPT 2006*, pages 37–53. Springer Berlin Heidelberg, 2006. doi:10.1007/11935230_3.

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Security evaluation report on SHA-224, SHA-512/224, SHA-512/256, and the six SHA-3 functions. Technical report, CRYPTREC, 2015.

[DR02] Douceur and John R. The Sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag. Accessed November 30, 2019. URL: http://dl.acm.org/citation.cfm?id=646334.687813.

[Esp19] Espressif Systems. *ESP8266EX Datasheet*, 2019. Version 6.2. Accessed November 30, 2019. URL: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[FP18] João Ferreira and Miguel L. Pardal. Witness-based location proofs for mobile devices. In *17th IEEE International Symposium on Network Computing and Applications (NCA)*, November 2018.

[FS14] Kassem Fawaz and Kang G. Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*. ACM Press, 2014. doi:10.1145/2660267.2660270.

[GKRT14] Sebastien Gambs, Marc-Olivier Killijian, Matthieu Roy, and Moussa Traore. PROPS: A PRivacy-preserving location proof system. In *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*. IEEE, oct 2014. doi:10.1109/srds.2014.37.

[Gooa] Google LLC. Indoor Maps – About. Accessed November 30, 2019. URL: https://www.google.com/maps/about/partners/indoormaps/.

[Goob] Google LLC. Location and Context APIs. Accessed November 30, 2019. URL: https://developers.google.com/location-context/.

[Gooc] Google LLC. Privacy: MAC Randomization - Android Open Source Project. Accessed November 30, 2019. URL: https://source.android.com/devices/tech/connect/wifi-mac-randomization.

[GSJ17] A. Gómez, M. Server, and A. J. Jara. Turismo inteligente y patrimonio cultural: un sector a explorar en el desarrollo de las smart cities. *International Journal of Scientific Management and Tourism*, 3(1):389–411, 2017.

[GSXK15] Ulrike Gretzel, Marianna Sigala, Zheng Xiang, and Chulmo Koo. Smart tourism: foundations and developments. *Electronic Markets*, 25(3):179–188, aug 2015. doi:10.1007/s12525-015-0196-8.

[HB01] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001. doi:10.1109/2.940014.

[HER] HERE Technologies. HERE Indoor Positioning. Accessed November 30, 2019. URL: https://indoor.here.com/.

[IEE] IEEE standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks—specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. doi:10.1109/ieeestd.2016.7786995.

[Int19] International Data Corporation. Worldwide Global DataSphere IoT Device and Data Forecast, 2019-2023, 2019. Accessed November 30, 2019. URL: https://www.idc.com/getdoc.jsp?containerId=prUS45213219.

[KFK+14] Katharina Krombholz, Peter Frühwirt, Peter Kieseberg, Ioannis Kapsalis, Markus Huber, and Edgar Weippl. QR code security: A survey of attacks and challenges for usable security. In *Lecture Notes in Computer Science*, pages 79–90. Springer International Publishing, 2014. doi:10.1007/978-3-319-07620-1_8.

[KK17] Dongwook Kim and Sungbum Kim. The role of mobile technology in tourism: Patents, articles, news, and mobile tour app reviews. *Sustainability*, 9(11):2082, nov 2017. doi:10.3390/su9112082.

[KZHH14] Rasib Khan, Shams Zawoad, Md Munirul Haque, and Ragib Hasan. OTIT: Towards secure provenance modeling for location proofs. In *Proceedings of the 9th ACM symposium on Information, computer and communications security - ASIA CCS '14*. ACM Press, 2014. doi:10.1145/2590296.2590339.

[Lan02] Marc Langheinrich. A privacy awareness system for ubiquitous computing environments. In *UbiComp 2002: Ubiquitous Computing*, pages 237–245. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-45809-3_19.

[LGD+15] Xingxing Li, Maorong Ge, Xiaolei Dai, Xiaodong Ren, Mathias Fritsche, Jens Wickert, and Harald Schuh. Accuracy and reliability of multi-GNSS real-time precise positioning: GPS, GLONASS, BeiDou, and Galileo. *Journal of Geodesy*, 89(6):607–635, mar 2015. doi:10.1007/s00190-015-0802-8.

[MFD03] G. Myles, A. Friday, and N. Davies. Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing*, 2(1):56–64, jan 2003. doi:10.1109/mprv.2003.1186726.

[Mic] Microsoft Corporation. Windows.Devices.Geolocation Namespace - UWP App Developer — Microsoft Docs. Accessed November 30, 2019. URL: https://docs.microsoft.com/en-us/uwp/api/Windows.Devices.Geolocation.

[MMD+17] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C. Rye, and Dane Brown. A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies*, 2017(4):365–383, oct 2017. doi:10.1515/popets-2017-0054.

[MMPR11] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. TOTP: Time-Based One-Time Password Algorithm. RFC 6238, RFC Editor, May 2011. Accessed November 30, 2019. URL: https://www.rfc-editor.org/rfc/rfc6238.txt.

[MSLK14] Alex T. Mariakakis, Souvik Sen, Jeongkeun Lee, and Kyu-Han Kim. SAIL. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14*. ACM Press, 2014. doi:10.1145/2594368.2594393.

[OA15] Kenneth Olmstead and Michelle Atkinson. Apps permissions in the Google Play Store, 2015. Accessed November 30, 2019. URL: https://www.pewresearch.org/internet/2015/11/10/an-analysis-of-android-app-permissions/.

[Oto] OtoSense Inc. OtoSense. Accessed November 30, 2019. URL: http://www.otosense.com/.

[PAA+16] Angelos Pillos, Khalid Alghamidi, Noura Alzamel, Veselin Pavlov, and Swetha Machanavajhala. A real-time environmental sound recognition system for the Android OS. In *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[RFA+13] Mirco Rossi, Sebastian Feese, Oliver Amft, Nils Braune, Sandro Martis, and Gerhard Troster. AmbientSense: A real-time ambient sound recognition system for smartphones. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, mar 2013. doi:10.1109/percomw.2013.6529487.

[SE15] Abdul Rehman Shahid and Amany Elbanna. The impact of crowdsourcing on organisational practices: The case of crowdmapping, 2015. doi:10.18151/7217474.

[Str19] Strategy Analytics. Global Connected and IoT Device Forecast Update, 2019. Accessed November 30, 2019. URL: https://www.strategyanalytics.com/access-services/devices/connected-home/consumer-electronics/reports/report-detail/global-connected-and-iot-device-forecast-update.

[TCB12] Manoop Talasila, Reza Curtmola, and Cristian Borcea. LINK: Location verification through immediate neighbors knowledge. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 210–223. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-29154-8_18.

[ZBC+14] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, feb 2014. doi:10.1109/jiot.2014.2306328.

[ZC11] Zhichao Zhu and Guohong Cao. APPLAUS: A privacy-preserving location proof updating system for location-based services. In *2011 Proceedings IEEE INFOCOM*. IEEE, apr 2011. doi:10.1109/infcom.2011.5934991.