Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Réseaux, Télécommunications, Systèmes et Architecture

Présentée et soutenue par :

M. MOUSSA TRAORE

le mardi 7 juillet 2015

Titre :

PROTOCOLES DE SECURITE POUR ETABLIR LES DISTANCES ET AUTHENTIFIER LA POSITION POUR LES APPAREILS MOBILES

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Laboratoire d'Analyse et d'Architecture des Systèmes (L.A.A.S.)

Directeur(s) de Thèse :

M. MARC-OLIVIER KILLIJIAN

M. MATTHIEU ROY

Rapporteurs :

M. PANOS PAPADIMITRATOS, KUNGLIGA TEKNISKA HOGSKOLAN STOKHOLM

M. SEBASTIEN CANARD, ORANGE LABS CAEN

Membre(s) du jury :

Mme ISABELLE GUERIN LASSOUS, UNIVERSITE LYON 1, Président

M. ABDELMALEK BENZEKRI, UNIVERSITE TOULOUSE 3, Membre

M. MATTHIEU ROY, LAAS TOULOUSE, Membre

M. PHILIPPE GABORIT, UNIVERSITE DE LIMOGES, Membre

M. SEBASTIEN GAMBS, UNIVERSITE RENNES 1, Membre

Contents

Introduction générale

La géolocalisation consiste en la capacité d'un système de communication à déterminer la position géographique d'un équipement (*i.e.* un téléphone portable par exemple). Ce procédé est utilisé par les services géolocalisés afin de fournir à l'utilisateur des informations utiles (intérêts de visite, plans, météo) relatives à sa position géographique.

Les applications de la géolocalisation sont nombreuses mais il a été démontré dans [GKdPC10] qu'elles présentent un réel danger en matière de respect de la vie privée de l'utilisateur. Différents textes de loi ont été adoptés dans le but de réglémenter l'usage des données localisation. Ainsi, la directive [PC] traite de la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données au niveau de la communauté européenne. La directive 2002/58/CE vient préciser cette dernière en posant des règles spécifiques sur les traitements des données à caractère personnel et la protection de la vie privée dans le secteur des communications électroniques. En France, cette protection est assurée par la Loi n 78-17 du 6 Janvier 1978 relative à l'informatique, aux fichiers et aux libertés.

Le but de cette thèse est donc d'apporter des primitives de géo-communication qui permettent la construction d'applications géolocalisés respectueuses de la vie privée de l'utilisateur. L'approche suivie est à la fois architecturale, algorithmique et cryptographique afin de limiter la dissémination d'information d'ordre privé. Cette thèse est structurée comme suit:

- Le **Chapitre 2** introduit le cadre général des travaux présentés dans cette thèse, c'est-à-dire la localisation et ses enjeux. Plus précisément, ce chapitre débute par une présentation des principales techniques de localisation et les principales attaques qui ont été développées contre elles. Ensuite, nous énonçons le problème de la vérification de la localisation avant de décrire succintement les techniques de résolution s'y afférant.

- Le **Chapitre 3** introduit les systèmes de preuve de localisation suivit d'une discussion sur les différentes architectures liés à ces services. Nous y proposons aussi une catégorisation de ces services.

- Le **Chapitre 4**, introduit un modèle de sécurité pour les systèmes de preuves de localisation.

- Le **Chapitre 5**, présente l'état de l'art des protocoles délimiteurs de distance.
- Le **Chapitre 6**, décrit VSSDB (*Verifiable Secret-Sharing based Distance-Bounding protocol*) notre première contribution scientifique. Il s'agit d'un protocole délimiteur de distance basé sur la cryptographie asymmétrique. Nous y analysons par la même occasion les propriétés de sécurité de ce protocole.
- Le **Chapitre 7**, décrit PROPS (PRivacy-preserving lOcation Proof System) notre deuxième contribution scientifique. Il s'agit d'un protocole de preuve de localisation respectueux de la vie privée conforme aux spécifications introduites dans le Chapitre 4.
- Le **Chapitre 8**, conclut ces travaux et introduit les perspectives.

Les travaux présentés dans ce manuscript ont fait l'objet de trois publications scientifiques à savoir: [GKaT12] qui introduit le *locanyme* comme primitive de base pour la protection de la vie privée dans les applications géolocalisées; [GKRT14] qui présente PROPS, notre système de preuve de localisation; [GKL$^+$] qui décrit VSSDB notre protocole délimiteur de distance.

Chapter 1

General introduction

Since the dawn of the Web, accessing the World Wide Web and the services it proposes, has mainly be experienced through the use of personal computers (PC) connected to each other with fixed wire connections. At these early times, the stationary nature of PC combined to the fixed nature of the connections has limited the access to Internet to customers present at their home or office only. Potential users away from these locations were unable to benefit from the web and its associated services. With the recent advances in the domain of positioning technologies such as Global Positioning System (GPS), Global System for Mobile Communication (GSM), Radio Frequency IDentification (RFID), and WiFi (802.11b/g/n) and the widespread deployment of wireless local area networks, we have witnessed the introduction of mobile devices equipped with geolocated and wireless communication capacities. One of the main advantage of this situation is the possibility for people to access Internet form anywhere they are then getting more out of existing web-based services in the form of Location-Based Services (LBS). Therefore, more and more Internet based services are personalizing the experience of their users by moving to a mobile setting, relying upon mobile networks and Wifi-enabled devices to deliver users with personalized services that are tailored to the current location of the individual querying the service.

LBS have been rapidly adopted by users because of the ubiquity and convenience they offer. They provide mobile users with a wide range of useful information such as weather forecasts, tourist attractions, landmarks, restaurants, gas stations, repair shops, public transportation options (including schedules) and proximity services that inform users when they are close to each other, are just a few examples of the types of information that would be more appropriate if tailored by the user's location. Using the location of users also allows sophisticated services like security systems to grant access based on establishing a user's presence within a specific area. LBS can also be employed for mobile commerce (m-commerce) and mobile advertisements. For example, a customer may query a specific product, and ask for all businesses in the area selling it. If the database includes other product information, such as prices and other terms, then real time comparison becomes feasible. Also, in many e-commerce situations the buyer and seller need to meet before validating the transaction, then the specific location of the buyer and seller becomes useful to the transaction. More surprisingly, cellular operators are beginning to offer different rates based on the location of callers, (*e.g.* in a designated home area).

Information regarding a user's location can be gathered in two main ways: *self-location* and *remote-location*. In the case of a self-location, the location data is supplied to the service by the user himself. For example, the user can input his location as a postal code but it is common that the user's location is automatically determined by his device before being transmitted to the LBS. One simple example in which self-location can be used is a user asking for nearby cinemas or requesting weather forecast in his city. In this situation, the user first provides the LBS with his location, then the server looks into its database for cinemas nearby the desired location. There are many different techniques that can be used for automatically determine the location of a user. Some are more appropriate for outdoor environments while some others are better for indoors locations and some offer good results in both environments. For more details about these positioning systems, we refer the reader to Chapter 2. For self-location, as the location information is supplied directly by the user or a device under his control, the information provided cannot be guaranteed to be accurate. Therefore, the LBS cannot have high level of trust in such information. In fact, even if the GPS device is contained within a Tamper-Resistant Module, the resulting location is not guaranteed as GPS signals can also be interfered from outside the security module (spoofing attack). These spoofed signals can lead to the computation of an incorrect location, leading to the provision of a false location to the LBS. Also, remark there is not a mean to check for the freshness of a GPS coordinates, so data can be replayed or transferred from one device to another.

In the situation called remote-location, an external entity is responsible to position the user, rather than the user providing its location directly to the service. The external entity might not only be a single entity but a group of collaborating entities exchanging information to determine with precision the position of the user. In order to accomplish this, the user must carry a device that can be accessed by the entity. An example of such a situation would be a mobile telephony network. Through the use of triangulation and other techniques, mobile phone companies can compute the location of a user within their network, provided that the user remains within the area of the network. This method of locating a user is somewhat more difficult to tamper with, but some simple approaches can be employed. An example of such an approach is the use of a Faraday cage to block or restrict the signal emitted from the user's device. However, this form of tampering requires some form of physical interference with the device, which may not always be viable. Therefore, while remote-location is vulnerable to interference, it is not as susceptible as self-location. The primary weakness of remote location is its reliance on the infrastructure to provide a location. This reliance limits the use of any remote location system to those areas containing an infrastructure and restricts applicability of the system based on the use of specific devices.

As self-location cannot be relied upon to provide trustworthy location information and remote-location requires adherence to a specific device or area in which infrastructure exists, a new approach must be found. An alternative solution is to redefine the problem. Rather than attempting to locate a user's device, the user's device makes a claim regarding its location that can then be verified. By altering the approach taken, the problem of providing a trustworthy location becomes much more manageable. The issue of locating the user is eliminated and the focus is placed on the veracity of the claim. More specifically, rather than a system attempting to prove the location of the user, the user provides proof of his claim. This approach prevents the need for special and costly hardware or an overly complex solution. To date, similar to the case of remote location, the trend in localization techniques relies upon a pre-existing infrastructure to provide trusted devices with which claiming devices can interact. However, reliance upon a fixed or limited infrastructure reduces the applicability of localization technology. In addition to this, the cost of employing the system increases dramatically. With mobile devices and mobile networks becoming a staple of current technology, the need for a specific infrastructure of devices is no longer present.

This thesis presents research on the development of a decentralized system for the verification of location claims, designed without the requirement of a pre-existing infrastructure for use as proof providers. Instead, our system relies on the collaboration of the users present at the same location to generate proof shares as a testimony of their presence at a geographical position. Then, a verification service can extract a location proof from the location proof shares collected by a user.

The development of the location verification system presented here, namely PRivacy-preserving lOcation Proof System (PROPS), is achieved through the combination of two main approaches. The first approach presents a novel distance-bounding protocol [GKL$^+$], based on an asymmetric setting and offering resistance to last known localization attacks. The second approach introduces a new security protocol designed to protect the gathering of location proof by mobile device making a location claim.

The thesis is structured as follows:

- In **Chapter 2**, we introduce the issues of localization and location verification before discussing existing approaches for solving this problem. In particular, we outline several important localization techniques employed within the field. We discuss a selection of existing schemes, provide context and background information for the research presented in this thesis. We also identify the primary attacks existing on localization techniques.

- In **Chapter 3**, we provide more details about location proof systems. More precisely, we first introduce the location proof systems before categorizing the related work in the field and then discussing the pros and cons of the two main approaches.

- In **Chapter 4**, we discuss the security properties that are desirable for such protocol and outline the design process undertaken to achieve a protocol supporting these properties.
- In **Chapter 5**, we present the state-of-the-art on distance-bounding protocols.
- In **Chapter 6**, we describe our first contribution to the area of private localization services: VSSDB, a Verifiable Secret-Sharing based Distance-Bounding protocol.
- In **Chapter 7**, we present our proposal for a privacy-preserving location proof system called PROPS for a PRivacy-preserving lOcation Proof System. We also analyze the security of our protocol using the security framework introduced in Chapter 4.
- In **Chapter 8**, we conclude the present work and give future research direction.

The research presented in this work has been published in the proceedings of three international conferences. [GKaT12] gives an early overview of the location verification system. [GKRT14] presents PROPS and its extensions in an earlier form. It also briefly discusses the security properties of the protocol. Our work on distance-bounding is presented in [GKL$^+$], with discussion and analysis about the security it offers against localization attacks.

Chapitre 2: État de l'art

Ce chapitre a pour but d'introduire le cadre général des travaux présentés dans cette thèse, c'est-à-dire la géolocalisation, ses usages ainsi que les problématiques sous-jacentes liées à la protection des données personnelles et à la vérification de la position annoncée par les utilisateurs. Nous définissons la géolocalisation comme l'action permettant de déterminer la position d'une personne, ou plus précisément d'un objet le caractérisant. Elle est, en pratique, rendue possible grâce aux techniques de positionnement (*positioning techniques*) telles que: la triangulation, latération ou le fingerprinting. Ces techniques permettent d'obtenir les coordonnées d'un point par rapport à plusieurs autres connus sur une carte.

La géolocalisation était, il y encore quelques années, réservée aux professionnels et spécialistes du transport maritime, terrestre et aérien. Aujourd'hui de nombreuses applications y font appel, y compris pour les particuliers: ce sont les services basés sur la localisation. Les services basés sur la localisation ont fait leur apparition depuis le milieu des années 2000 et ont très rapidement été adoptées par le grand public. Google est un des premiers grands noms de l'informatique à avoir intégré la localisation dans ses applications pour terminaux mobiles. Par exemple les résultats des recherches sont différents en fonction de l'emplacement géographique du client. Plus encore, de nombreuses applications sportives et de loisirs utilisent la localisation en temps réel de leurs utilisateurs afin de générer des informations précises sur les lieux, les vitesses et les obstacles pour mieux se repérer.

Comme toute nouvelle technologie, la géolocalisation mais surtout les service qui en font usage apportent leur lot d'inquiétudes. En effet, de nombreux services géolocalisés utilisent les données de localisation recoltées de leurs utilisateurs à des fins commerciales, de profilage, etc. De plus, toute entité pouvant accéder à ces informations peut alors les utiliser pour déduire des informations personnelles sur les individus concernés par ces données. En effet, grâce aux attaques par inférence, il est possible de déduire leur domicile ou lieu de travail, prédire leurs possibles localisations futures, identifier leurs point d'intérêt (POI), extraire la sémantique des POIs les plus fréquentés ou même de construire leur réseau social, provoquant ainsi une violation de la vie privée des utilisateurs. Pour remédier à ce problème, différentes techniques d'anonymisation des données personnelles ont été dévéloppées par la communauté scientifiques. Ces techniques bien qu'ayant été prouvé éfficaces, ne sont aujourd'hui pas intégré dans la vaste majorité des applications géolicalisées. En complément de ces initiatives techniques, des textes de lois ont été adoptés dans l'Union

Européenne et aux États-Unis afin de définir un cadre juridique et limiter l'usage qui est fait des données personnelles de localisation des particuliers.

D'un point de vue sécurité, les services basés sur la localisation présentes des limites. Ceci est d'autant plus important dans des applications basés sur l'authentification ou le contrôle d'accès à distance. Pour remédier à ce problème, il existe deux grandes familles de protocoles de sécurité proposées par la communauté scientifique. Il s'agit des protocoles délimiteur de distance et les systèmes de preuves de localisation. Ces familles de protocoles sont discutées dans le détail aux Chapitres 3 et 4.

Chapter 2

Background

2.1 Introduction

Recent advances in ubiquitous connectivity, as well as mobile and embedded systems, have led to the development of a plethora of services that are personalizing the services they deliver according to the location of the user querying the service. These services known as Location-Based Services (LBS) offer many advantages to consumers. For instance, a LBS can be used for resource discovery (*e.g*, finding the closest restroom from my position), path-finding (*e.g.*, computing the shortest route to a gas station), real-time social applications (*e.g.*, informing me about the presence of my friends in the vicinity) or location-based gaming (*e.g.*, playing with the nearest challenger). This chapter provides the context and relevant background on the topic of localization and privacy. In the first part of the chapter we define the concept of positioning (cf. Section 2.2) with the techniques used to compute the position of a device. Next, we introduce positioning systems (cf. Section 2.3) and location-based services (cf. Section 2.4). Afterwards, in Section 2.5, we discuss the notion of privacy in general and how it should be handled in the context of localization. Thereafter, we discuss technologies that can be used to protect privacy of user (cf. Section 2.6). Finally, Section 2.7 introduces the main research topic presented in this work: the secure location verification problem and the two major techniques that have been proposed in the literature to resolve it, namely: distance bounding and location proof system.

2.2 Positioning

2.2.1 What is positioning?

Positioning is defined as the act of determining a user's location, with user's location taken to mean a point on the earth's surface, which could be absolute or relative. The absolute location is designated using a geographical coordinates system (*e.g.*, latitude, longitude and altitude). In constrat, the relative location uses a place of reference to locate the desired location (*e.g.*, Toulouse is in the South-West of France, here Toulouse is located relative to France). In the literature, many terms are used for positioning like localization, location finding, position location, geolocalization or location sensing. Currently, users as physical entities can only be located relative to the devices they carry due to the current limitations of technology. Therefore, positioning is in fact the act of determining a device's location, under the assumption that a specific user is located in proximity to that device. In the following, we present some localization techniques used to obtain the location of a user.

2.2.2 Positioning techniques

Localization of a node is estimated through communication between localized nodes (i.e: *anchors*) and unlocalized node for determining their geometrical placement or position. Location is determined by means of distance and angle between nodes. There are many concepts used in positioning a node such as the following.

Fingerprinting technique.

Fingerprinting is a technique that uses location dependent characteristics of the radio signal to estimate the distance separating the unlocalized nodes and the reference anchors. The set of location dependent data is called the *fingerprint*. To better understand the logic behind fingerprinting, let us consider a local area network equipped with Wi-Fi access points. For each AP, the strength of the Wi-Fi signal emitted progressively decreases as we move away from the emitting point. Therefore, measuring the received signal strength from an AP can be used to estimate the distance to this access point. Location fingerprinting is composed of two phases : the offline one and the online one. During the online phase, a radio map is constructed by measuring the fingerprint on different points of the location site with their location coordinates. Then, the map is recorded into a database. During the offline phase, when the system receives a location query from an unlocalized node, an algorithm tries to match the node's recorded fingerprint values with the ones on the radio map that was previously created to infer the exact location of the node. In practice, the accuracy of fingerprinting can be enhanced by exploiting different methods such as: probabilistic methods [RMT$^+$02], k-Nearest-Neighbour [PVA00], neural networks [YLAU11], support vector machine, and smallest M-vertex polygon [VJS$^+$07], Bayesian inference [HPALP09] and others [HPALP09].

Trilateration technique.

Trilateration is based on the measurement of the propagation time of a signal to estimate the distance of an unlocalized node relative to at least three anchors. The relative positions of the unlocalized node can be obtained by measuring either the *Time of Arrival* (ToA) or the *Round Trip Time* (RTT) of the signal. With ToA, the anchor emits a time-stamped signal towards the localization site. When the signal arrived at an unlocalized node, the distance between the anchor and the node is calculated from the transmission time delay and the corresponding speed of the signal. The knowledge of the ToA allows the node to be situated on a circle (for simplicity, we confine the coordinates plan to two dimensions and ideal transmission conditions) with the emitter at the center. By adding two transmitters, the node can be located at the intersection point of the three circles. The "tri" in trilateration is because we need at least three emitting anchors to obtain a precise position.

A derivation of ToA is the Round-Trip Time (RTT) or *Round-Trip Time of Flight* (RToF). RTT measures the time of flight of the signal pulse travelling from the anchor to the unlocalized node and back. While ToA requires two local clocks in both nodes, RTT uses only one node namely the anchor to record the transmitting and arrival times. RTT is usually used in the design of distance bounding protocols. We discuss distance bounding and its role in secure localization in Section 2.7.

Trilateration is a highly accurate technique. However, it requires high processing capability and accurate synchronization between the nodes and the emitting anchors. The Global Positioning System (GPS) uses such technique to position a device relative to a constellation of satellites orbiting the earth.

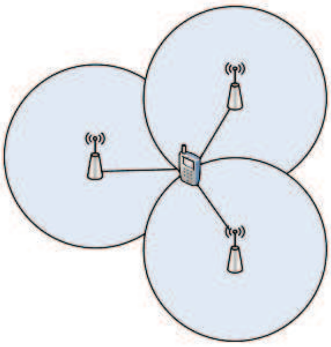

Figure 2.1: Trilateration technique.

Multilateration technique

Multilateration is a localization technique based on the measurement of the difference in distance to pair of anchors that broadcast signals at known times. The distance between anchors of a pair is known and they are synchronized to exchange a signal. An unlocalized node that listen to a pair exchanging messages, can measure the Time Difference of Arrival (TDoA) of the signals at its position. TDoA allows to compute the difference in distance to the two emitting anchors [FNI13]. Here, in the same spirit as trilateration, the possible locations of the unlocalized node are found on a hyperbola (2D) or a hyperboloid (3D). To locate the exact location along that curve, multilateration relies on at least four measurements: a second measurement taken to a different pair of emitting anchors will produce a second curve, which intersects with the first, etc. While the curves overlap, a small number of possible locations are revealed, producing the position of the node.

Triangulation technique.

Triangulation allows a unlocalized node to calculate its position by measuring two directions towards two reference anchors. Since the positions of the anchors are known, it is hence possible to construct a triangle where one of the sides and two of the angles are known, with the unlocalized node at the third point. This information is enough to define the triangle completely and hence deduce the position of the node. Triangulation uses the *Angle of Arrival* (AoA) of radio signal at the unlocalized node. AoA can be measured using several antennas placed side by side (an array of antennas) or a directional antenna to measure the phase difference between the signals received by the antennas. For better positioning accuracy, the angles of incidence of at least three signals may be used for triangulation.

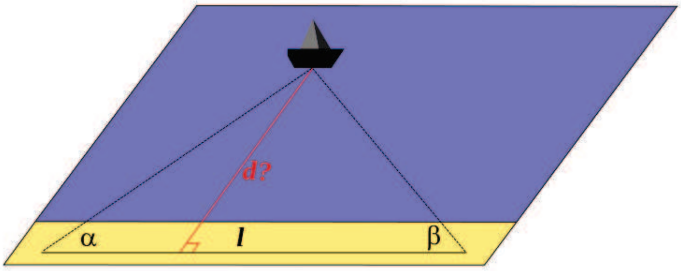

Figure 2.2: Trilateration technique.

2.2.3 Localization attacks

This attack denotes any action intended to defeat the correctness of the localization process. This can be done for instance by influencing the outcome of a positioning device, in fact if an adversary knows the positioning techniques used by his device, then he can launch a specialized attack to influence the outcome of the positioning device. In the case that the positioning device is used as a black-box by the adversary, he can try to directly attack the LBS. In the following, we enumerate the six existing localisation attacks.

Denial of Service (DoS): The goal of a DoS attack [HHP03] is to prevent a specific device or set of devices within an area from localizing others or being localized by the system. This is generally achieved by interfering with the communication medium. For example, with RF-based positioning systems the attacker tries to interfere with the targeted devices ability to receive and send messages (signals), usually by flooding, injecting or jamming the communication channel with messages of its own. This prevents those devices present in the location targeted by the attacker from receiving or sending messages properly, thus rendering them incapable of localizing themselves properly.

Emitter impersonation: This attack consists in impersonating a signal transmitter in front of a targeted device [TRPČ09]. It exploits the fact that some fingerprinting-based systems relies on access point MAC addresses for the identifications of the anchors. Such an attack has been demonstrated as effective against the SkyHook's localization system in [TRPČ09].

Signal properties attacks: These are attacks targeted against the properties of the signal used in the localization process. We can distinguish between *attenuation* and *amplification* attacks [CTM07]. In an attenuation attack, specialised equipment, such as an absorbing barrier, is used to decrease the speed and strength at which a signal can propagate. This makes appear as though the signal has travelled farther, thereby tricking the unlocalized node into believing that the anchor is farther than it truly is. Inversely, in an amplification attack, a malicious device may employ specialised equipment, such as a directional antenna with an extended range, to achieve localization at a location closer to the anchor devices than it truly is. Amplification attacks can impact localization results in which the signal strength of the signal is employed (see for instance the fingerprinting based localization technique 2.2.2). However, due to their time-based approaches, triangulation and multilateration based systems are not as easily defrauded by such attacks.

Proxy attack: In this attack, a malicious node records messages generated within his range (by the system), before "tunneling them" them to another area in which is located either another node under its control or a colluding agent within the network [Mar12]. "Tunnelling" involves transmitting the recorded messages directly from one agent to another, without forwarding through multiple nodes. Then the colluding agent while physically located out of communication range from the localization anchors, can still communicate with them and being localized as close. Remote-location systems are particularly vunlnerable to such an attack.

Collusion attack: In this attack, a malicous device attempts to influence its own localization results by having a colluding device masquerade as the device being localized and transmit signals from a closer position to the receivers. The receivers therefore calculate the location of the colluding device, rather than that of the device being localized, thus providing a malicious device with a false location in closer proximity to the receivers. This form of attack is effective primarily due to the fact that localization techniques localize the source of a signal, not a specific device.

Sybil attack: This attack has a similar impact to the collusion attack, but with only a single attacker involved. In this attack, a system believes itself to be employing several different devices in a localization, when in reality it employs only one [SW07]. This is achieved through an attacker creating multiple identities for itself and attempting to pass them as individual devices, a simple task without the presence of some central authority registering all devices. When a central authority is employed, an alternative method may be used to circumvent this measure, in which the attacker steals the identity of another device within the system. This approach may not be detected if the attacker prevents the legitimate owner of that identity from communicating with the central authority, or if the owner is not active at that point.

2.3 Positioning systems

A positioning system is a mechanism for determining the location of an object in space. Technologies for this task exist ranging from worldwide coverage with meter accuracy to workspace coverage with millimeter accuracy. Positioning systems can be grouped in many different ways, including indoor versus outdoor systems or cellular versus sensor network positioning. In this section, we categorize positioning systems depending of the environment targeted. Indeed, we may distinguish: indoor localization systems [Dem03], outdoor localization systems and hybrid localization systems.

2.3.1 Outdoor positioning systems

The most famous positioning system of this category is undeniably the Global Positioning System (GPS). GPS employs trilateration to calculate the position of a device relative to a constellation of satellites orbiting the Earth. Due to the fact that GPS coordinates can be forged and signals spoofed, the United States military has created a similar system to GPS known as the Precise Positioning Service (PPS) which encrypts all signals to prevent spoofing. However this technology is unavailable to civilians. Galileo is the first public alternative to GPS. The first stage of the Galileo programme was agreed upon officially on 26 May 2003 by the European Union and the European Space Agency. Galileo constitutes a significant improvement compared to GPS. Indeed it has an increased level of resistance to jamming and will include encryption within its signals to prevent spoofing attacks. Another well known outdoor positioning system is the Assisted GPS (A-GPS). A-GPS was introduced on request of the U.S. FCC's 911 [DR01], to make location data of an emergency caller available to the corresponding public safety service [DR01]. A-GPS is extensively used with GPS-capable cellular phones with Internet connection. In practice, the accuracy of outdoors positioning systems decrease significantly when employed within highly built up areas. Therefore, alternative approaches to position in closed area are required. For more details about existing outdoor positioning systems, we refer the reader to [BCKM04].

2.3.2 Indoor positioning systems

An indoor positioning system considers only indoor environments. Dempsey [Dem03] defines it as a system that continuously and in real-time determines the position of something or someone in a physical and closed space such as shopping mall, hospitals, airport, subway and university campuses, etc. In general, many of those systems are based on short range wireless technologies such as InfraRed (IR), UltraWideBand (UWB), Bluetooth, etc.

The first major indoor positioning system proposed was the ActiveBadge [WHFaG92], created by researchers from AT&T Cambridge. The system was originally designed to track objects within a specific area. It functions through the use of a wearable device that periodically emits its unique identity over IR every fifteen seconds to a fixed infrastructure of receivers. The beaconing message emitted by the device bounces off (but does not escape) the walls bounding the room, thus ensuring the signal fills the room. The receiver fixed within the room receives the message and can infer that the device is located within its walls. ActiveBadge provides symbolic location information of each device such as the room in which the active badge is.

Researchers at AT&T Cambridge provides 3-D position and orientation information for tracked tags: the ActiveBat system [APR+99], an improved approach to ActiveBadge which instead of IR employs UltraSound (US) and RF. Within ActiveBat, many *slave* anchors are disseminated over the localization area with a *master* anchor capable of transmitting an RF signal. When a node need to be localized, the master anchor transmits a RF signal to the node as a signal to begin the localization process. Upon reception of the RF signal, the unlocalized node responds with an US signal, which is heard by the surrounding slaves. The nearby anchors, which have also received the initialisation signal sent by the master anchor, record the TDoA between the RF signal and the US signal. From this difference, each slave can extract the distance between their location and that of the unlocalized node and send this information to the master. Finally, the master combines the TDoA from the slaves at known locations and calculate the location of the node using multilateration. Despite addressing some issues within the ActiveBadge system, ActiveBat is heavily dependant on a centralized approach with a fixed infrastructure, thus limiting its applicability. The system, as reported in 1999, can locate Bats to within 9 centimeters of their true position for 95 percent of the measurements,

In contrast, the Cricket system [PCB00], has been developed to decentralize the localization process and reduce overheads. Though Cricket also employs a combination of US and RF signals. In particular, the system operates in the same spirit as the ActiveBat at the difference that the node being localized is responsible to compute its own location. More precisely, pairs of anchors regularly emit RF messages indicating their location along with US signal. The node that is listening to the exchanges in his vicinity can measure ToA and consequently calculate its location by itself using triangulation. For a complete overview and comparison of indoor positioning system and their accuracy, we refer the reader to [GLN09].

2.3.3 Hybrid positioning systems

Hybrid positioning systems are localization systems that can work both in indoor and outdoor environments. They usually combine different sources of metadata and signal such as GPS data, cellular signal, WiFi signal, Bluetooth sensors, IP address, network environment data, etc. Hybrid positioning systems are useful for certain civilian and commercial services that need to work well in urban areas in order to be commercially and practically viable. These systems were specifically designed to overcome the limitations of GPS, which is very exact in open spaces (precision from 1 centimeter to 100 meters [BCKM04]), but works poorly indoors or between tall buildings because GPS signals get absorbed by the buildings. By comparison, signals emitted by cellular tower are not hindered by buildings or bad weather, but they provide less precise position (between 250 meters and 35 kilometers [BCKM04]).

With the rapid growth of wireless access points in urban areas, exploiting those wireless networks may give very exact positioning (between 10 meters and 100 meters in outdoor and 3 meters in indoor [BCKM04]) provided that the localization area has high Wi-Fi density and that a comprehensive database of the access points is available. The access point database is created using a technique called *wardriving*. Wardriving refers to the practice of searching a target area for Wi-Fi networks using a portable computer while travelling by car (derivations are *warwalking* and *warbiking*). When the *Personal Digital Assistant* (PDA) detects a wireless network, it collects the MAC address with the SSID of the access point, and associates this with the location (usually determined by GPS) and the signal strength of the detected network. Then, it uploads the entire record to a database. When an unlocalized node wants to request its location, it submits (usually via an encrypted channel) a list of the MAC addresses of Wi-Fi access point identified in its range to the database, then the known positions of one or more of these access points is retrieved, allowing the node to compute its location using fingerprinting. Hybrid positioning systems include SkyHook's, Google Maps and Navizon.

2.4 Location-based services

A Location-Based Service (LBS) is an application that tailors the service provided to the current geolocated context. LBS uses the consumer's location provided by a positioning system to provide its service. The popularity of LBS is rapidly growing due in part to the increased use of location-enabled devices like smartphones and tablets. LBS offers a variety of services such as navigation tools to help one reach a destination (*e.g.*, Google Maps); local search to help you find nearby businesses or events (*e.g.*, Yelp); friend-finders and social networking (*e.g.*, Tinder); applications that allow you to "check in" at certain locations (*e.g.*, Foursquare); and applications that can link your location to activities (*e.g.*, Facebook and Twitter).

LBS can further be classified based on the functionality it provides. We distinguish between four categories of LBS:

Geo-targeting applications: These are applications designed to respond to the following question: “Where am I?” Given the position of a user, equipped with a device with geolocated capabilities, this LBS shows on a map or in a social network application, the location of the user or some of the content that he has posted, such as "Bob was at Blagnac Airport". For instance, Picasa, Foursquares, Facebook places are examples of geo-targeting services.

Recommendation services: They provide users with suggestions about products or places near him. This type of LBS can be proactive or reactive. Examples of proactive recommendation services include geo-advertising, which pushes information to users about products or services when they reach a certain area, such as a coupon for a pizza when the user is at Capitol square (*e.g.*, PayPal Media Network or Twitter) or proposal to pedestrians potential means of transportation in their neighbourhood whose destination match their own destination [GKC+13]. In contrast, a reactive recommendation service queries about the nearest place to find some service or product such as "where is the closest Peruvian restaurant?".

Navigation system: This category of LBS includes application to compute a trajectory to go from one point to another, (*e.g.*, the best itinerary to go from the Charles-de-Gaulle airport to the Eiffel tower). Google Maps and Waze are examples of such services. Some navigation systems like in the AMORES project [ADG+12] and CooVIA [1] can notify if there is friends of the user in the neighbourhood or itinerary to pick up passengers in order to carpool.

Monitoring systems: They continuously record the movements of users in order to report crowded zones in a city for statistics or to maintain the flow of road traffic. Examples of such application are MapCity, Tom Tom, TeleNav and the density analyzers like Skyhook or Sense Network.

Other LBS classifications exist such as the one presented by Andersen and Kjærjaard, who divides monitoring category in crowd-sensing and city watch [AK12] or Sidney Shek who classifies LBS depending on the target market, application type and technical capabilities [She10]. Many users actually access LBS through mobile phones, but any location-aware devices such as laptop, desktop computers, tablets, and in-car navigation systems and assistance systems can also be used to access these services. However, this is not without an impact on privacy.

[1] `https://coovia.fr/`

2.5 Privacy and LBS

Privacy taken as a single term cannot be formally defined as it means many things in different contexts and culture. For example, opening a door without knocking might be considered a serious privacy violation in one culture and yet permitted in another. In 1968, Alan Westin defined privacy as being: “the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others” [Wes68].

Currently, there is no consensus on a single definition of privacy; in fact, privacy varies across domains and can therefore be considered as being a subjective notion. For example, in computer security, privacy is related to the confidentiality of a message or a file, protected in general through encryption. Thus, the attack will attempt to decrypt the ciphertext in order to extract the information it contains. In secure multi-party computation [Gol98], privacy is associated with the information one is able to infer besides the result of the computation of some protocol. Therefore, the objective of the attack is to learn as much as possible from the protocol, by reading the messages passing among the nodes, by having a malicious behaviour, by implementing a man-in-the-middle attack or by colluding with several nodes. In social network, privacy is defined by the amount of information (messages, photos, applications, location, etc.) one shares with his friends. Therefore, the objective of the inference attack is to profile members of the network based on their features by extracting implicit information from the social graph.

In the new era of information technology, privacy can be defined as the right to keep control over our *sensitive information*. Sensitive information corresponds to any data related to our person like birth date, personal phone number, location and our religious affiliation. Although privacy is considered a fundamental right, it is difficult to give a precise definition of the term. The domain of privacy partially overlaps security, including for instance the concepts of appropriate use, as well as protection of information.

However, one thing that everyone agrees to is that privacy is a fundamental building block of a robust democracy; but protecting privacy, when personal data is shared, whether voluntarily or not, is a difficult and complex task.

2.5.1 Location privacy

Location privacy [BS03] refers to the ability to prevent other parties from learning one’s current or past location. Indeed, LBS requires to know the location of a user so they can offer various useful services. However, each time a user transmits his location information to the service for a single purpose, it may be stored or combined with other information to produce a history of the user’s activities or a more detailed profile for advertising or other purposes. Thus, it is necessary to extend the concept of privacy of an individual to a spatio-temporal context; in other words the sensitive data to protect is the position at a given time.

So far, few positioning systems have considered privacy as an initial design criterion. That is for instance the case for the Active Badge system that detects the location of each node and broadcasts the information to everyone in the building. The system was originally deployed assuming anyone in the building was trustworthy. This inconvenient has been addressed in the Cricket system in which the flow of information recorded by the unlocalized node, can directly be used to compute its own location (*self-positioning*) without the system notifying its presence thus location privacy is preserved. This approach is also used in GPS.While application designer have a key role in the location privacy protection of users, they are not the only responsible of privacy leakage; Indeed, some persons disseminate publicly, almost in real-time, their current location via popular (geo)social application such as Facebook places, Foursquare or Twitter. In turn, this data can be collected to predict whether or not they are currently at home like with the website Please rob me[2]. Thus, there are many players in the mobile space who have a contributing role to ensuring end-to-end privacy, whether it is the *device manufacturer*, the *operating system* and *platform developer*, *network providers*, *application developers*, *data processors* and even *users* themselves.

Cottrill [Cot11] proposed to allocate roles based on who is responsible for providing protection to such a data. More precisely, the suggestions goes as follow: users should establish their own privacy policy through the available privacy settings. They should refuse to use an application without a declared privacy policy and, have the minimum amount of location data if possible. Users should be conscious about the risks of location disclosure. While companies have a *privacy-by-disaster* approach that is they only care about privacy protection after a significant incident is made public [oNC11]; their developers should be initiated to *privacy-by-design*. Privacy by Design is a concept developed back in the '90s [Cav11]. It aims to identify and examine possible privacy breaches (*i.e.* data protection problems, etc.) when designing a technology in order to incorporate privacy protection mechanisms into the overall design, instead of having to come up with "patches" later on. If systems are built without respect for user privacy, and adhere to only the most basic requirements, companies may suffer when their systems get compromised or when a large base of customers reacts negatively. Privacy by Design was unanimously recognized and supported by the International Assembly of Data Protection and Privacy Commissioners, and is a recommended approach in major U.S. and European regulatory efforts [oNC11]. Service providers are also responsible for setting up access controls to guarantee information confidentiality and secure storage to prevent information leak. Telecom operators should follow a self-regulatory approach in order to preserve the privacy of their clients. Finally, public and private agencies should provide regulation and guidelines to the data processors about how to manipulate location data in order to ensure location privacy protection. In the following, we introduce some of the privacy regulations that have been introduced for privacy protection.

[2] `http://pleaserobme.com/`

2.5.2 Privacy regulation

Privacy is recognized as a fundamental right by governments and organizations all over the world. This situation has caused the adoption of new regulation laws in order to regulate the transmission and sharing of user sensitive information. More precisely, in 1995, the European Union has adopted the 95/46/CE Data Protection Directive [PC] to harmonize the laws of members states on the protection of individuals with regard to the processing of personal data and on the free movement of such data. This directive established several basic principles for European citizens. These principles include the following rights:

- The right to know where the data originated,
- The right to have inaccurate data rectified,
- The right of recourse in the event of unlawful processing,
- The right to withhold permission to use data under some circumstances.

On January 2012, the European Commission unveiled a new draft legislative package to establish a unified European data protection law [95412]. This regulation brings new privacy rights, including data subject's *right of portability* and the *right to be forgotten*, for citizen in the EU. The right of portability will allow a transfer of all data from one provider to another upon request, for example transfer of a social media profile or email, whereas the right to be forgotten will allow people to wipe their history clean. The other remarkable changes to the 95/46/CE Data Protection Directive are the following:

- the EU data protection regulation will also apply for all non-EU companies without any establishment in the EU, provided that the processing of data is directed at EU residents,
- any processing of personal data will require providing clear and simple information to concerned individuals as well as obtaining specific and explicit consent by such individuals for the processing of their data (Opt-in), other than in cases in which the data protection regime explicitly allows the processing of personal data,
- the processing of data of individuals under the age of 13 will in general require parental consent, which will make it more difficult for companies to conduct business aiming at minors,
- all companies will be obligated to notify EU data protection authorities as well as the individuals whose data are concerned by any breaches of data protection regulations or data leaks without undue delay, that is within 24 hours,
- a harsh sanction regime will be established in case of breach of the unified EU data protection law allowing data protection authorities to impose penalties of up to 2 % of a company's worldwide turnover in case of severe data protection breaches.

Moreover, some standards and privacy risk analysis related to privacy are currently being put forward. For example, the ISO/IEC 29100 [ISO] about Information technology, Security techniques and Privacy framework, the AICPA/CICA Privacy Risk Assessment Tool, NIST Privacy Assessment, and the Methodology for Privacy Risk Management. In U.S., the Location Privacy Protection Act of 2014 [Fra], introduced by Senator Al Franken, targets those apps designed to maliciously track individuals without their knowledge. The bill stipulates that each individual may give his explicit consent before using a service, the collecting entities, the collectable data and its usage. The bill does not include location data stored locally on the device (the user should be able to delete the contents of the location data document periodically just as he would delete a log document). The bill that was approved by the Senate Judiciary Committee, also requires mobile services to disclose the names of the advertising networks or other third parties with which they share consumers locations.

2.6 Privacy solutions

A number of technical solutions have been proposed to protect the privacy of LBS users. In the sequel, we present the major ones to be used throughout the manuscript.

2.6.1 Anonymisation

The aim of anonymisation is to ensure that the LBS will not be able to link requests to specific users. A mean to comply with this, is by using *anonymiser*. Anonymiser is an intermediate trusted system that groups together k users requests before sending them to LBS provider for process, and returns specific responses to users. The value of k can be configured to ensure sufficient anonymity [Liu, 2009]. The main drawback to anonymisers is that they rely on the anonymiser being a trusted system, thus it becomes a single point of failure issue in the architecture.

2.6.2 Cryptography

Cryptography [Sch94] is the science of writing in secret code and is an ancient art; some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. Cryptographic techniques such as encryption and hash chain are commonly applied to conceal information and to hide user's identities. These techniques may require more computational power and sophisticated application code than other methods. Cryptographic techniques would be useful for both identity-driven and pseudonym-driven applications. For example, in [HWK04] authors propose blind signatures as a mean to protect the identity of a user from his communications provider. Privacy-enhanced cryptographic primitives include group signature [CS97, FZ12], zero-knowledge proof of knowledge protocol [FFS88], hash chains [LMP11] and commitment schemes [BCC88].

2.6.3 Location granularity

Location information has a defined accuracy depending on the technology used. For example, GPS may be able to resolve location down to 5-10 meters while GSM-based localization resolve it to 250 meters to 35 kilometers [BCKM04]. While a high-accuracy technology may be used, applications may not need such high resolution, so a client application can reduce the accuracy of its location when sending requests to LBS provider thereby reducing the possibility that the LBS provider can identify the user's exact location. Accuracy filtering can be done by selecting the appropriate location technology via the smartphone's configuration parameters for accuracy, or by post-filtering of the location information. One example of post-filtering was a technique described in [LMP11], in which hash chain is used to certify the location information in such a way that the hashes of the chain correspond to an increasing level of precision or in [LH10] in which five levels of granularity are encrypted and inserted in the request, then LBS requests the corresponding key to the user to unveil the granularity that it needs.

2.7 The secure location verification problem

LBS are services that take advantage of the position of users to deliver service tailored to their geolocated context. By the use of a positioning device, a user can acquire his current position and then transmits it to LBS for processes. However, as pointed in [SW09], a malicious user can lie about his position, by having his device transmitting a location of his choice and thus access services inadequately. This situation is critical for emerging applications such as real-time traffic monitoring, location-based access control, discount tied to the visit of a particular shop or local electronic election. Thus, the enforcement of location-aware security policies (*e.g.*, this laptop should not be taken out of this building, this file should not be opened outside of a secure room, etc..) requires trusted location information. So, one of the first question that naturally arises from this is how can we convince others about our current position? Thereafter, we will refer to this problem as the Secure Location Verification Problem. Secure location verification has been addressed in the literature in two different ways: using time-based approaches, namely *distance-bounding protocols* (DB) [BC93] and *location proof systems* [SW09]. DB is a timing-based approach that allows an entity (*the prover*) to interact with a set of authorities (*verifiers*) in order to prove his current position. Location proof system is an architecture that allows a prover to prove his past position to a set of verifiers. Many location proof systems are based on DB protocol to defeat proxy attacks and terrorist fraud attacks. In the following, we review the state of the art in the domains of distance-bounding protocol and location proof systems.

2.7.1 Distance-bounding protocol

Distance bounding protocols are cryptographic protocols that enable a verifier V to establish an upper bound on the physical distance to a prover P. They were originally introduced by Brands and Chaum [BC93] to defeat relay attacks and are based on Round Trip Time techniques (see Section 2.2.2) which measuring the delay between sending out a challenge bits and receiving back the corresponding response bits. The delay time for receiving the correct responses enables V to compute an upper-bound on the distance from the prover, as the round trip delay time divided into twice the speed of the radio wave. Both radio frequency (RF) and ultrasound channels have been used in the design of secure and lightweight distance bounding protocols. However, since the speed of sound is six order of magnitude lower than the one of light (the propagation of RF in air approaches the in-vacuum speed of light), it is established that RF is more secure.

There exists a significant literature on distance-bounding (DB) protocols. Usually, DB protocols is ran in association with a lightweight authentication protocol. This authentication procedure can either be symmetric (*i.e.*, based on a common secret shared between the prover and the verifier [RNTS07, TP07, ALM11, KAK$^+$09, BMV13b]) or asymmetric (*i.e.*, dependent of a secret that is only known to the prover [BC93, BB05a]). DB protocols are made of three main phases which are: the *initialization phase* (or *first slow phase*) during which the prover and the verifier pre-computes the values that are needed to run the DB protocol. Thereafter, the first slow phase is followed by the *interactive phase* (or *fast bit exchange phase*) during which the prover and the verifier exchange bits at very high speed. Finally, there is the *verification phase* (or *final slow phase*), that allows to determine if the protocol succeeds or not.

The security level of these protocols can be evaluated according to their resistance to five types of attack, namely: *slow phase impersonation* [AT09a], *distance fraud* [BC93], *mafia fraud* [Des88], *distance hijacking* [CRSC12] and *terrorist fraud* [BBD$^+$91]. Each of these attacks targets a specific aspect of the protocol such as the authentication or the distance estimated by the verifier. In an impersonation, the adversary tries to impersonate a legitimate prover to the verifier but does not lie on its distance. In contrast, in a distance fraud a legitimate but malicious prover attempts to lie on his distance to the prover. A mafia fraud is an attack in which an adversary defeats a DB protocol using a man-in-the-middle attack between the verifier and an honest prover located outside the area. In a distance hijacking attack, a malicious prover takes advantage of a protocol executed between a honest prover and the verifier. Finally, terrorist fraud denotes an attack in which the adversary plays the man-in-the-middle between the verifier and a prover that is not located in the vicinity. However in contrast to the mafia fraud, the prover is also malicious and willingly cooperate with the adversary to help him achieve his objective of defeating the DB protocol. However, the malicious prover only helps the adversary to the extent that he does not have to give him his long term secret in the process.

Over the years, DB protocols have matured to the point that the MIFARE Plus RFID tag [NXP11] is the first commercial product (at least up to our knowledge) implementing distance bounding in a commercial product. Distance bounding can be used as a building block for the design of other security systems like it is the case of location proof systems.

2.7.2 Location proof systems

A location proof system designates an architecture that allows a prover to obtain a certificate of his actual position as an evidence of his presence in this geolocated context [SW09]. We call the certificate obtained by the prover: the location proof (LP). In the same spirit of real life alibi, a LP can be used by its owner to prove that he was located at position *pos* at time *time* (as endorsed in the LP) to a remote service or to authority. We call the entity that checks the validity of a location proof the *verifier*. A secure location proof verification system requires the following two properties:

- *Correctness*: if the prover and the verifier behaves honestly, and the prover was indeed at the location claimed then the verifier will accept the claimed position *pos* of the prover using a location proof.

- *Soundness*: if V behaves honestly and accepts the position claimed by a prover, then the prover or one of colluding party is located at the position accepted.

Location proofs systems are made of two main phases: the proof gathering phase and the proof verification phase. The proof gathering phase is ran by the prover each time he needs to collect a proof for the location in which he is physically present. The physical presence of the prover at the claimed location can be established using distance-bounding protocol with a node of the system (*witness*). Once the witness is convinced of the presence of the prover in his proximity, it outputs a location proof to the latter containing the current location and time informations.

The heart of any location proof system is the gathering process that can broadly be categorized into two classes depending on whether a user collects his location proof (1) through an interaction with non-mobile and dedicated access points deployed by the system administrator, we call this the centralized gathering approach or (2) by directly interacting with other users of the system which are located in his surrounding, we call this the collaborative gathering approach. In the former approach, proof issuers are Wifi-Access points advertising their presence by regularly broadcasting beacon signals to their surrounding area. Afterwards, any nearby device can collect these beacons as a prerequisite to the process of creation of the location proof by the Access Point. The benefits of a collaborative gathering approach over a centralized gathering approach are multiples. In fact, the system is more scalable, deployment/maintenance cost are reduced, pollution is also reduced and finally it offers better resistance to DoS attacks.

The second main phase of location proof system is the verification phase. This phase allows a location-based service (verifier) to be convinced of the provenance of the prover before he can access the service. The goal of this phase is for the prover to provide the verifier with location proof containing the correct spatio-temporal information (past or present) as needed for the functioning of the location-based service.

2.8 Conclusion

In this chapter, we addressed the general concept of positioning, the techniques commonly used by positioning systems and the attacks that can be launched against them. We also described the functioning of location-based applications that are built on top of positioning systems and how their uses can rapidly becoming a threat against the privacy of users. Then, we have briefly introduced the privacy regulations in Europe and USA. In addition, we also described the techniques such as anonymisation, cryptography and location granularity that have been developed so that only minimal quality of location data needs to be given to LBS providers. Then, we introduce the concept of secure location verification and the two majors countermeasures developed in the literature to improve the security of location-based services, namely location proof system and distance bounding protocols. In the next chapters, we will give more details about such countermeasures and position our contributions to location verification problem. More precisely, in the next chapter, we will review in more detail the concept of location proof systems.

Les preuves de localisation

L'apparition de l'informatique mobile et la démocratisation des dispositifs de communications mobiles (GSM, PDA, smart gadgets...) amènent les organisations à ouvrir leurs systèmes d'information afin de le rendre disponible n'importe où et n'importe quand. Ceci ne peut se faire sans une prise en compte de la dimension mobile des utilisateurs dans le modèle de sécurité des accès. Ainsi, un système d'information doit dorénavant être capable de prendre en compte des caractéristiques contextuelles (telles que la position du requérant et l'heure) pour garantir un contrôle d'accès fiable à certaines ressources protégées. Dans ce chapitre, nous abordons la thématique des preuves de localisation. Une preuve de localisation désigne un certificat numérique attestant de la position géographique d'un individu (le *prouveur*), plus précisément de son téléphone, à un instant donné. Les preuves de localisation sont obtenues par le biais d'une requête émanant du prouveur et à destination d'un système de preuve de localisation.

Un système de preuve de localisation désigne une architecture logicielle permettant à un prouveur d'obtenir des preuves de localisation. En pratique, chaque preuve de localisation est émise par une tierce-partie agissant alors comme un *témoin* pour le prouveur.

De nombreux systèmes de preuves de localisation ont été proposées dans la litterature. Nous les décrivons dans le détail à la Section, puis nous les catégorisons en deux grandes familles. La première famille regroupe les système basée sur la présence d'une infrastructure de points d'accès et la seconde famille regroupe les systèmes basées sur la coopération entre les différents utilisateurs. La différence réside dans la manière dont les prouveurs sont autorisés à récupérer des preuves de localisation. En effet, dans les systèmes basés sur les points d'accès, chaque prouveur ne peut demander une preuve de localisation que lorsqu'il se trouve à proximité de point d'accès spécifiques agréés par le système. Par contre, dans les sytèmes basés sur la coopération, les utilisateurs présents dans une même zone géographique peuvent directement communiquer et se générer mutuellement des preuves de localisation.

Nous discutons aussi des impacts que peuvent avoir l'une ou l'autre ces deux architectures. Il en ressort que les systèmes basés sur les points d'accès souffrent de leur dépendance à une infrastructure spécifique. En effet, ils passent difficilement à l'échelle, nécessitent des coûts d'entretien et d'administration. De plus, du point de vue de la vie privée, les utilisateurs de tels systèmes ne sont pas à l'abri d'un traçage permanant. À l'inverse, les systèmes basés sur la coopération exploitent les ressources grandissantes de nos smartphones. Cependant, elles nécessitent la mise en place de mécanismes strictes pour éviter les abus liés aux collusions entre utilisateurs.

Dans le chapitre qui suit, nous discuterons des propriétés de sécurité et de vie privée que nous jugeons indispensables aux systèmes de preuves de localisation en général et aux systèmes basés sur la coopération en particulier. Nous verrons en particulier que la sécurité de la phase de collecte des preuves de localisation peut être significativement améliorée en faisant appels aux protocoles délimiteurs de distance.

Chapter 3

Location proof systems

3.1 Introduction

Location-Based Services (LBS) are a general class of services that use location data to control features. Individuals can use it to find nearest interesting spots. Parents can use it to locate their child at any time. Professionals can locate their vehicles, track personnel, deliveries and detect any problem. Authorities can use it to manage suspect alibis. However, as discussed in the previous chapter, LBS cannot rely solely on the user's devices to discover and transmit location information because they have an incentive to cheat. Instead, such applications may require their users to prove their locations: location proof system serves this purpose.

This chapter introduces the domain of location proof systems. We describe the related work by grouping them according to the way location proof are collected by the users before discussing the pros and cons of the different approaches.

3.2 State-of-the-art

A location proof system designates an architecture that allows its users to prove their location to location-based services before they are authorized to access a service. A *location proof* designates the piece of information that is actually used by the users to convince the content provider about their current or past location.

To illustrate the importance of location proof systems, consider the scenario of a content delivery server (*i.e.* a movie server). The movie server wants to restrict the content it delivers to users depending on their locations due to some copyright laws. Before starting a download, the server needs first to ensure that the recipient device is authorized to access the downloadable content in accordance with the copyright laws. Thus, the mobile device needs to obtain a location proof before requesting the hosted service. Therefore, one can imagine that the mobile device transmits to the movie server the identifier of its nearest communicating cell tower. Then, using the latter identifier and the help of the telephony network, the movie server can infer the position of the mobile user by mapping the identifier of the cell tower to its geographical location. The movie server can then verify the device's current location and then decide whether or not access to the content should be granted. In another application, suppose a store wants to offer discounts to frequent customers. In this context, making devices aware of their location is not sufficient; instead, users must be able to show evidence of their repeated visits to the store.

In the sequel, we investigate the literature on location proof system by grouping them according to the way users obtain location proof. Firstly, we enumerate system in which users collect location proof by interacting with access point disseminated over the localization area. We call such approach of implementing location proof system, the *bipartie gathering approach*. Secondly, we consider systems in which users collect location proof by collaborating together. We call this latter approach the *collaborative gathering approach*.

3.2.1 Bipartite gathering approach

This is the first class of location proof system introduced at a time when the wireless networks were becoming increasingly popular, and the first Wifi-enabled mobile devices were introduced to the market. In such systems, user requests location proof from designated Wifi Access-Point (AP) present over the localization area. The protocols were designed to be expanded on-the-shelf to existing Wifi-networks. The expansion is done by a system administrator, responsible to configure each AP of the network to run a simple program which computes location proof transmitted to the requester devices. In the following, we propose to describe the systems which have been developed on this idea [WF03, SW09, LH10].

Waters and Felten (2003)

Waters and Felten [WF03] turn their attention to equip small wireless network with mechanisms that can vouch for the presence of mobile devices within the small area they cover.

Here, the access-point responsible to issue location proof is called the *LM (Location Manager)*. To obtain a location proof, user convinces the LM that they are within proximity. LM judges of their proximity to user by using RTT like techniques.

To start the process of obtaining location proof, user needs to know the public key of the service he want to convince of his position. Once this condition is satisfied, the user sends a location proof request to the nearest LM. A location proof request is a simple message containing the user's public key encrypted with the public key of the service the user wants to convince of his position. The request is then transmitted encrypted with the public key of the intended LM. Authors did not explicitly describe how user can obtain the public key corresponding to the nearest LM.

After the reception of the proof request, the LM starts its clock and sends a nonce to the user. At the reception of the nonce, the user must send back the nonce to the location manager as soon as possible. When the LM receives the response from the user, it stops its clock and computes the latency of the round-trip exchange with the user. Next, the LM issues a location proof to the user. The proof contains two blocks of information. The first block contains the public key of the LM, the latency computed by the LM, the current date and time information and lastly the encryption of the user's public key. The second block contains a signature of the first block with the private key of the LM.

At the reception of the location proof, the user must immediately sends a message for proving his location to the remote service, he has specified in the proof request. The message contains two blocks of information, the first one contains the location proof received from the LM concatenated with the LM's public key concatenated with the user's public key. The second block contains a signature of the first block with the private key of the user to express the consent of the user to prove his position to the service. The message is then sent to the service encrypted with its corresponding public key for approval. Service has access to a database mapping the public key of LM to their physical position. Then by retrieving the latency of the communication between the LM and the user, the physical position of the LM, the service can decide of the validity of the user's location claim.

With respect to privacy, the main objective of this protocol is to hide the identity of the device and the verifier from LM and potential eavesdroppers. This protocol is not secure against attack in which a malicious prover transmits his public key and a well defined location proof request to a colluder to masquerade as him in front of an AP. Another limitation is that each location proof collected by the prover can be used with only one verifier.

Saroiu and Wolman (2009)

Saroiu and Wolman [SW09] have proposed to include the latitude and longitude coordinates of Wi-Fi access points into the beacon frames they periodically broadcast to announce their presence. Then, a device capturing such beacon, can use it to explicitly request a location proof from that respective AP. A location proof request contains the sequence number of the last beacon captured from the AP and the public key of the user. The proof request is also signed with the private key of the user before it is sent to the AP.

At the reception of a proof request, the AP checks two things. The first one is that the sequence number included in the request matches well with a beacon it has broadcasted a few times ago. The second thing is that the signature of the proof request corresponds to the public key of the user. If those verifications succeed, the AP replies with a location proof addressed to the user. The proof contains two blocks of information. The first block contains in clear text: the access point's public key, the device's public key, the date and time of the creation of the proof and the latitude and longitude coordinates of the AP. The second block of information contains a digital signature of the first block made with the secret key of the AP.

To prove his location to a service, a user signs his location proof and prepends to it his public key. The resulting message is then transmitted to the service. Upon reception of the message, the service performs two signature verification. Firstly, it checks the user's signature over the location proof using the public key of the user. Secondly, it checks that signature included in the second part of the location proof is correct. Those verifications ensure that the location proof has not been tampered while being transmitted to the service and that the client has not tampered with the location proof created by the AP. If those verifications succeed, then the service is convinced that the client is indeed the genuine recipient of the location proof and that the location proof is legitimate.

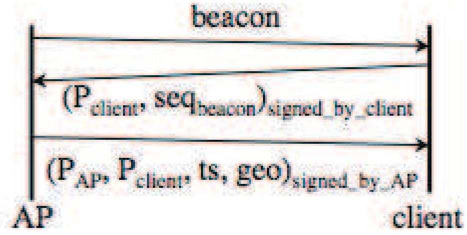
beacon
(P_client, seq_beacon)_signed_by_client
(P_AP, P_client, ts, geo)_signed_by_AP
AP
client

Figure 3.1: Saroiu and Wolman Location proof System.

This protocol is not secure against an attack in which a malicious prover transmits his public key and a well defined location proof request to a colluder to help the latter in masquerading as him in front of an AP. However, the authors have also propose a number of improvements to enforce privacy guarantees of their protocol and strong authentication. From a privacy point of view, one of the main issue of this protocol is that prover needs to publicly reveals his identity before obtaining a location proof.

VeriPlace (2010)

VeriPlace has been proposed by Luo and Hengartner in [LH10]. The system puts in relation a number of trusted-third party that are the *TTPU (Trusted Third Party for managing User informations*, the *TTPL (Trusted Third Party for managing Location)* and the *CDA (Cheating Detection Authority).* The above mentioned trusted entities are run by different parties to avoid collusion and to protect user's privacy: each entity knows either a user's identity or her location, but not both of them.

Within VeriPlace, the first step is for users to request intermediate proof to AP present in their vicinity. To obtain intermediate location proof from AP, user first sends a message to the TTPU to obtain a token. The message contains the identity of the user and the identity of the AP encrypted with the CDA public's key. In response to this message, the TTPU replies to the user with a nonce. At the same time, the TTPU registers in his database the identity of the user, the encryption of the identity of the AP and the date and time at which it has created the nonce for the user.

Afterwards, the user uses the token to obtain intermediate location proof from the AP. An intermediate proof has two blocks. The first block contains the public identity of the AP, the token from the TTPU and the time the access point received the proof request from the user. The second block contains a digital signature of the first block with the private key of the AP.

Later, to provide a service with a location proof, user first presents his intermediate proof to the TTPL to obtain a final proof. The user also specifies a granularity for the proof he wants to receive. User can choose among five levels of granularity for the location they want to reveal. The TTPL creates a final location proof by replacing the AP identity in an intermediate proof with the location of the AP at the granularity specified by the user and the AP's signature with its own signature.

When verifying a final location proof submitted by the user, the service submits the identity of the user and the date and time information to the TTPU. The TTPU extracts a list of encrypted access point identities from his database that match the user identity and roughly the same date and time values. The list is then submitted to the CDA for cheating detection.

After receiving the list of encrypted AP information from the TTPU, the CDA decrypts it and checks whether any two APs of the list are far apart, which is a sign of cheating (because the same user can not request location proofs at two far-apart places simultaneously). The CDA notifies the TTPU in case cheating is detected.

If all the steps above are passed successfully and no cheating is detected by the CDA then the service accepts the proof.

3.2.2 Cooperative gathering approach

In the cooperative gathering approach, users obtain location proof not by communicating with a dedicated infrastructure but by collaborating with other users present at the same location. This approach assumes that the users can communicate with each other using range communication technology, such as Bluetooth or Wi-Fi in Adhoc mode.

The SLVPGP system (2009)

SLVPGP stands for the *Secure Location Verification Proof Gathering Protocol*. Graham and Gray propose three versions of the protocol with increasing security properties in [GG09]. In the following we describe the functionality of the third version that offers the best security properties among the others. The protocol is based on the assumption that there exists a central server knowing the identities and public keys related to all users of the system. It is also assumed that for any location claim made by a user, the central server knows the identity of some other devices present at the location.

At the beginning of the protocol, the user sends his location and his identity to the central server for verification. The central server replies with a list of other users known to be present in the same area. At the reception of the list, the user contacts each of the device present in the list to obtain a proof of proximity from them. More precisely, the user provides each device with a message containing their identity and two nonces: the initiating nonce and the replying nonce. These nonces are used by both parties to compute two registers: the register of challenges and the register of responses.

Next, each device sends to the user a message containing a value picked at random in the register of challenges, the position of the value in the register and a nonce. Then the device starts its clock. At the reception of the previous message, the user checks that the value received exists at the position specified in the register of challenges. If the verification succeeds, the user continues the protocol and replies with a message containing a value picked at random in the register of responses, the position of the value in the register of responses and the nonce received from the neighbouring device. Once the user's response is received, if the value received from the user is present within the register of response, the neighbouring device sends to the user a proof of proximity message containing, its identity, its position, the round-trip latency and the date and time information of the creation of the message.

Finally, the user sends all the proximity messages to the central server. Based on those informations, the central server computes a location proof for the user. The location proof obtained from the central server can now be used to prove his location to other services.

APPLAUS

Zhu and Cao designed APPLAUS [ZC11] that stands for A Privacy-Preserving LocAtion Updating System.

When joining the system, each user receives a set of public/private key pairs and the corresponding public key certificates from an online trusted authority called the Certification Authority (CA). The public part of each key pair is considered as a pseudonym for the user, the private part serves to create location proofs for other users and the public key certificate helps to authenticate the user. The table for mapping the pseudonyms of users to their real identity is secret and managed by the CA.

To obtain a location proof, a mobile user broadcasts a proof request containing his pseudonym at the given time (computed by a deterministic algorithm, see [ZC11] for more details), to its surrounding devices. Each surrounding devices decides whether to accept the location proof request according to its scheduling. In case of a positive response to the location proof request, the neighbouring device creates a location proof for the user. The location proof contains the public key of the device, the public key of the user, the current date and time, the GPS coordinates of the device and a signature on all the previous using the current secret key of the device. The location proof is then encrypted using the public key of the location storage server and transmitted to the user.

Finally, the user sends the encrypted location proofs to the location proof storage server using the Internet. During the verification process, the user announces to the verification service the time at which he was at position *pos*.Then, the verification service queries the CA for the location proofs of this specific prover. This query contains the real identity of the user and a time interval. The CA first authenticates the verification service, and then converts the real identity of the user to its corresponding pseudonyms during that time period and retrieves their location proofs from the location storage server. The location proof server only returns hashed location rather than the location in clear to the CA, who then forwards it to the verification service. Finally, the verification service compares the hashed location with the claimed location acquired from the user to decide if the claimed location is authentic or not.

LINK (2012)

The *Location verification through Immediate Neighbors Knowledge (LINK)* has been proposed by Talasila, Curtmola and Borcea. LINK uses a trusted centralized entity called the *Location Certification Authority (LCA)*, that is responsible to register user within the system. During registration, each user receives a public and private key pair, and a default trust score value. The LCA is responsible to update the trust scores associated to each user of the system based on their interactions. More precisely, a user's trust score is additively increased when her claim is successfully authenticated and multiplicatively decreased otherwise in order to discourage malicious behaviour. As a consequence, during the collection of a proof, user prefer to interact with neighbouring users with high trust scores.

Before submitting a location proof to a verification service, the users must register a proof request signed with his private key to the LCA. The proof request contains the identity of the user, the location claimed by the user, a sequence number to identify the request and the name of the service he wants to convince of his position. Then, the user broadcasts a location certification request to his neighbouring users. The request contains the identity of the user and the sequence number registered with the LCA. Next, each neighbouring device sends a message signed with their respective private key to the LCA. This message contains the identity of the neighbouring device, the location of this device and the certification request received from the user. The certification request is included to allow the LCA to match the user's proof request with the certification messages received from the neighbouring devices.

The LCA waits for a short period of time to receive enough certification messages and then starts the decision process. The LCA decides the claim's authenticity based on spatio-temporal correlation between the users and the trust score associated with each neighbouring users. Finally, the LCA informs the verification service about its decision, causing the verification service to provide or deny service to the user.

Privacy-preserving Alibi system (2012)

Davis, Chen and Franklin introduced ALIBI [DCF12]. Location proof can be collected from neighbouring users and from so-called *public corraborator*. A public corroborator acts like an AP in the bipartite gathering approach: their identities and locations are publicly known to the users of the system and they can vouch for the position claim by a user. At the registration phase, each user and public corroborator receive a public/private key pair.

A user can opportunistically request the creation of a location proof whenever public corroborators or other users are available in his neighbourhood. The user then creates an OwnerStatement, which is included in the location proof request. The OwnerStatement is tied to the identity of the user and his current location, but by itself cannot reveal those informations unless the user reveals that link. At the reception of the OwnerStatement, a neighbouring user signs it using his private key and commits to this data using a cryptographic string commitment scheme. Finally, the neighbouring user prepends to the commitment his own location and time informations to form the location proof. At this time, the only information that can be obtained from the location proof is the location of the neighbouring user and the time at which the proof has been created. Other information like the identity of the user, the identity of the neighbour and its signature, are hidden within the proof.

During the verification of the location proof, the user first contacts the neighbouring user to obtain the openings for the hidden part of the location proof and the public key of the neighbouring user. Then the user sends those information to the verification service along with the information needed to open the OwnerStatement. Finally, the verification service uses the informations provided by the neighbouring user to obtain and check the signature of the OwnerStatement. Then, the information provided by the user are used to check that the OwnerStatement includes the real identity of the user. Finally, if the location and time information provided by the user corresponds to the information included in the location proof then the proof is accepted, otherwise the proof is rejected.

STAMP (2014)

Recently, Wang and co-authors propose STAMP [WZP$^+$], in which co-located mobile devices mutually generate location proof for each other using bluetooth or WiFi in ad-hoc mode. Then the prover convince a verifier of his location by showing him the location proofs he received. stamp ensures the integrity of LPS (*i.e*, a prover cannot modify the data included in a LPS once generated), the non-transferability, and the anonymity of prover and witnesses generating the proof. Users have also the possibility to choose the granularity to reveal to a verifier. However, in contrast to Props, the LPSs are encrypted under the CA public key, thus the prover cannot check himself the validity of location information endorsed by the witness. During the verification stage, the verifier needs to contact the CA to validate a LP. The collusion detection algorithm is an entropy-based trust evaluation approach like the one used in APPLAUS. STAMP incorporates the Bussard-Bagga distance-bounding protocol [BB05a] as a countermeasure to the proxy attacks. The authors also provide a prototype implementation on the Android platform with an averaging running time of 8 seconds.

3.2.3 Comparison

We propose here to discuss the pros and cons for each of the approach that we have enumerated early.

Recall that within a cooperative gathering approach, a user who needs a location proof can request the other surrounding users to generate such location proof. At the beginning of the 2000's, with the low performance of cell phones, this was not the best choice for implementing a location proof system. However, things have evolved, our smart phones and tablets have now sufficient processor power and storage memory to run even complex security protocols and store gigabytes of information. As mentioned in the literature, even complex protocol such as STAMP can run within second on a smart phone. Therefore, leveraging the creation of location proof to our mobile devices is now a perfectly viable and feasible idea. However, when we give users the ability to create and store location proofs, we also give them more possibilities to cheat the system. As pointed by many of the related work, we can imagine situation in which users may deviate from the normal running of the protocol, to create location proof for requester who are not physically present at the same place for money or profit. Therefore, a location proof system based on a collaborative gathering approach may also include mechanisms to detect collusion of users. We give more details about this in the Chapter 4. In addition, this type of infrastructure does not need to manage or monitor any of these mobile devices, thereby drastically reducing management costs and privacy concerns.

Within a bipartite architecture, there is publicly known wifi-access point that can generate location proof for users. Here, the advantage is that all the proof providers are trusted by default, so there is no risk of collusion between the proof issuers and the users. However attacks like distance fraud and terrorist fraud are still possible and we need to design security mechanisms to counter these attacks. Here, the security of the whole system relies on the Wifi-Access points side. In fact, these access points constitute the system's central point of failure and require attention of the system administrators. Each time the Access point is relocated then it must be reconfigured to the new longitude and latitude to provide the valid location proof to requester. Access-point located in outdoors environment can be equipped with GPS capabilities to dynamic reconfiguration after movement. In the case of indoor environment, the system can be equipped with a proper indoor or hybrid localization system or reconfiguration can be done manually by the system administrator. Due to the cost in the maintenance of Access-point, bipartite approach should be preferred in applications which only require a small-scale deployment of infrastructure capable of handing out locations proofs. For example, a coffee store can start running a promotion promising a free drink to any customers that visited their store daily in the past week. A

Wi-Fi access point issuing location proofs is a simple and cheap way of implementing such a promotion. Similarly, a teacher can offer rewards to those students who never miss a class during the semester. With location proofs, students can collect them and submit them at the end of the semester to receive their reward. A sensible consideration is making sure that APs are configured with the correct location coordinates. While it is cheap to provision APs with GPS to routinely determine their location, most APs are situated in indoor environments in which GPS does not work fine. One way to overcome this complexity is to provide the AP with an additional configuration interface for administrators. To point a location proof-enable AP, the administrator initially takes the AP outdoors and runs a setup program using GPS to establish the AP's location.

Independently of the chosen architecture, the system is subject to privacy issues. In a cooperative approach, users interact with other users so they may have control over how much information they disclose to their surrounding.

3.3 Conclusion

In this chapter, we introduced the notions of location proof, location proof system and location proof architecture. We classified the state of the art to the nature of proof issuers. In fact, we distinguish between systems in which users interact with dedicated access-points and systems in which users collaborate together. In general, all these protocols acts globally on a similar manner. That is, they consists of a series of exchange between the mobile device and the proof issuers. Then the proof issuers generate a signature attesting of the presence of the user in his neighbourhood. Location proof systems provide LBS with a mean to verify the position claims of mobiles users before granting them access to a service. In the next chapter, we analyse the security and privacy properties brought by the aforementioned systems.

Chapitre 4: Un modèle de sécurité et de vie privée pour les système de preuves de localisation.

Arpès avoir parcouru l'état de l'art des systèmes de preuves de localisation, le but de ce chapitre est d'étudier les propriétés de sécurité et de confidentialité des données nécessaires à ces dits systèmes. Cette étude a abouti à l'identification de menaces à considérer lors de la phase conception de futurs systèmes. Nous illustrons chacune de ces menaces par des scénarios d'attaques précis. L'identification de ces menaces, nous a permis par le suite d'énumérer les propriétés désirables à une preuve de localisation. En effet, une preuve de localisation doit être garantir la *non-transférabilité*, la *non-forgeabilité* et la *résistante aux attaques de type MITM et ses dérivées*. Côté vie privée, une preuve de localisation doit pouvoir garantir *l'anonymat* de ses utilisateurs et la *non-chaînabilité* de leurs actions à travers le système. Plus encore, nous montrons qu'il est nécessaire de protéger la phase de vérification des preuves contre de potentiels vérifieurs malveillants dont l'objectif serait d'établir un profil du prouveur, ou de ré-utiliser la preuve de localisation de ce dernier à des fins personnels, etc. Ce dernier point, bien que s'inscrivant dans des scénarios d'attaques possibles n'avait pas encore été abordé dans l'état de l'art. Nous encapsulons ensuite toutes nos recommandations ansi que celles déjà présente dans l'état de l'art dans un modèle de sécurité unique.

Finalement, ce chapitre se termine par une analyse détaillée des protocoles issus de l'état de l'art dans ce nouveau modèle. Cette comparaison montre que l'ensemble des systèmes de l'état de l'art sont vulnérables aux attaques de localisation de type MITM et ses dérivées. À cet effet, le prochain chapitre introduira les protocoles délimiteurs de distance qui peuvent être utilisés pour sécuriser le processus de collecte des preuves de localisation.

Chapter 4

Security model for privacy-preserving location proof system

4.1 Introduction

There exists a plethora of location-proof systems that have been developed in the literature during recent years (see Chapter 3). These systems aim at proposing infrastructure for the gathering and the verification of location proof. Some are based on a bipartite gathering approach in which users interact with publicly known access point to obtain location proof. Other systems enable users to directly collaborate together to generate location proofs. The purpose of location proof is to avoid situation in which a prover can lie about his position to a verifier. Then location proof system may thwart attacks in which provers pretend they are in a location where they are not. Unfortunately, there is no clean definition of the properties which may fulfil a location proof system. In fact, each of the system we found in the literature comes with its own terminology, definition and notion of security. For example, with the system [ZC11], users privacy is preserved with respect to the final verifier but witness can easily obtain the identity of the prover. In another one [WZP$^+$], user's privacy means that user's identity needs to be hidden from witnesses. So behind the intuitive idea of what privacy and security needs, we have different interpretation of such properties across environment.

This Chapter provides to provide a unified framework to ascertain the security and privacy properties of location proof systems. In Section 4.2, we model location proof systems in terms of their algorithms and the users interaction within the system. Then, in Section 4.3, we introduce an adversary model qualifying the attackers of the system in terms of their capabilities and objectives. Section 4.4 and Section 4.5 introduce the security and privacy properties that must be provided by location proofs system. Finally, in the

Section 4.6, we give an analysis of the related work before concluding in Section 4.7.

4.2 Preliminaries

This section aims to describe our security model for location proof system. The model first describes the users and their interactions, the algorithm and how the adversaries interact to cheat the system. We also describe some oracles that can be used by an adversary to obtain information about the system.

4.2.1 Modelling of users and their interactions

As introduced early, a location proof system is a set of mechanisms with which mobile users can obtain location proofs from proof issuers and with which applications can verify the location claimed by users.

To use the system, users need to be registered by a trusted entity that we call the *Certification Authority (CA)*. More precisely, the CA is responsible for setting the system global parameters and creating credentials for the users of the system. Credentials are substitutes for the real identity of the user within the system. The CA is therefore the entity knowing the mapping between the real identity and the credentials for each user of the system. Credentials are used to establish communications between users, to create proof of their location and to tied location proofs to a particular user. Because of the central role played by the credential, it must be protected in order not to be easily shared among users. Otherwise it may be difficult to properly identify users and distinguish between their actions within the system. One way to achieve this is to relate credentials to high sensitive information of user like their account access number or their security social number. Thus, deliberately revealing his credential may cause great damages to the user.

While caring about privacy, the role of the CA may be supported by a second entity that we call the *Anonymity Lifter (AL)*. While it is obvious that the AL may be part of the CA, we strongly recommend to split the role among two different entities to avoid abuses from the CA. The role of the AL is limited to the capacity of unveiling the user's identity from a proof of location to disambiguates situation. However, the complete process of identifying a user based on his interaction within the system may always require active collaboration between the AL to retrieve the credential and the CA to identify the real owner of the credential.

The users of a location proof system may play different roles among *prover*, *witness* and *verifier*. The role of prover is played by any user who wants either to obtain a location proof for the position he is visiting at a given time or to prove his location to a verification service. We do not require that the prover, at the moment he is collecting a location proof, knows the identity of the verification service he will interact with. The only condition to obtain a location proof is that the prover *shares the same context* with other users of the system. In practice, sharing the same context means to be present at a location at the same time and we use the term context to uniquely associate a location to the time.

A witness is a user of the system who shares the same context as the prover. Obviously, a witness can either refer to an Access Point in the case of a bipartite gathering approach or to a neighbouring user in the case of a collaborative approach. Witness reacts to location proof request received from prover by generating a *location share*. A location share designates a piece of contextual information that testifies the presence of the prover nearby the corresponding witness. Location share acts like testimonials of witness telling *"I testify that I have been within proximity of such prover at such location at such time"*. In other words, a location share may uniquely associate a context as defined by the prover to the prover and a witness. At this time, the prover may collect location shares from as many witnesses as possible to strengthen his future location claim to *verifier*.

A verifier is an entity permitted by the user to verify his location claimed for some purposes. When this situation occurs, the prover emits a location claim to *verifier* as his approval to start the process of verification of his past or actual location with the verifier. A location claim contains a location proof and additional information needed to check the validity of the proof. A location proof is created using location shares corresponding to the location claimed by the prover and can be checked by the intended verifier. Finally, if the location proof is genuine and corresponds to the context that the prover and the verifier have agreed on, then the verification process succeeds. Otherwise it is rejected by the verifier.

4.2.2 Definition of the algorithms

Browsing the wide literature of location proof system, we can define a location proof system as the following suite of algorithms ($\mathsf{Setup}, \mathsf{URegister}, \mathsf{VRegister}, \mathsf{LTestify}, \mathsf{LProve}, \mathsf{Vf}$). These algorithms are the following.

- $\mathsf{Setup}(\lambda)$ This algorithm is ran only one time at the initialisation of the system by the CA. It takes as input a security parameter λ which is related to the size of the secret keys employed in the system. The algorithm's output is a couple of keys $\mathsf{spar}, \mathsf{ppar})$ with (spar representing the secret parameter of the system. This information will be kept secret by the CA or splitted among the trusted parties like the AL if it does exists, while ppar represents the public parameter of the system that is publicly available to users. It may help them to interact each other, to verify a location proof, etc.

- $\mathsf{URegister}(\mathsf{ID_u}, \mathsf{spar}) \rightarrow (\mathsf{sk_u}, \mathsf{pk_u})$ When a user wants to join the system, he needs to run this procedure together with the CA or an authorized Issuer to obtain his credentials $\mathsf{sk_u}, \mathsf{pk_u}$ in which $\mathsf{sk_u}$ represents his private key and $\mathsf{pk_u}$ represents his public key. The algorithm takes as input the identity $\mathsf{ID_u}$ of the user and spar the the secret key of the CA.

- $\mathsf{VRegister}(\mathsf{ID_v}, \mathsf{spar}) \rightarrow (\mathsf{sk_v}, \mathsf{pk_v})$ This algorithm serves to register verifier to the system. The algorithm takes as input $\mathsf{ID_v}$ the public identity of the verifier and spar the secret parameter of the CA. It outputs the tuple of keys $(\mathsf{sk_v}, \mathsf{pk_v})$, which respectively represent the secret key and public key functions of the verifier. This allows any device to communicate over secure and authenticated channel with verifier once it possesses that verifier's identity.

- $\mathsf{LTestify}(\mathsf{L}, \mathsf{T}, \mathsf{sk_p}, \mathsf{sk_w}) \rightarrow \pi_\mathsf{i}$ This is a distributed algorithm that is ran between a prover and a witness to obtain a location proof. The prover initializes the session using his secret key $\mathsf{sk_p}$ and the witness does the same with his secret key $\mathsf{sk_w}$. L represents the location data that the prover want to certify, T represents the time at which the witness certify to have seen the prover at that location. The output is a location proof share denoted as π.

- $\mathsf{LProve}(\{\sum \pi_\mathsf{i}\}, \mathsf{L_i}, \mathsf{T_j}) \rightarrow \pi_\mathsf{L,T}$ This algorithm is ran by the prover to generate a location proof from the location shares he has collected. The algorithm takes as input the location shares π_i that corresponds to the claimed context. The values L_i and T_j represent the location and time respectively at granularity i and j that the prover is willing to disclose to the verifier. The output is a fresh location proof $\pi_\mathsf{L,T}$ to be showed to the verifier.

- $\mathsf{Vf}(\pi_\mathsf{L,T}, \mathsf{L_i}, \mathsf{T_j}) \rightarrow \{0, 1\}$ This algorithm enables the verifier to check the validity of a location proof received from the prover. The algorithm verifies that the location and time granularity L and T corresponds indeed to the data enclosed in the location proof $\pi_\mathsf{L,T}$. It is possible to envision that if the procedure outputs Reject, the verification has failed so the verifier suspects a cheating attempt from the current prover, then it may forwards the fraud evidence (*i.e.* $\pi_\mathsf{L,T}$) to the Anonymity Lifter (AL) who has the capacity to reveal the identity of the cheater.

4.3 Threat model

Location proof provides prover with a trustworthy set of information to convince verifiers about they positions they are claiming. Therefore, the security of location proof system could be threaten in different ways depending on the way location shares are managed by the users of the system. To illustrate our position, imagine a user who is able to modify the context included in the location share without invalidating the location proof. Such user can make any verifier believes that he was at a place where he was not and thus render the whole location proof architecture useless. We can also imagine verifiers collaborating together to use location proof they received from a particular prover to infer information about his social network, etc.

As a consequence, the processes of creation, verification and storage of a location share has to be carefully designed so as to counter attacks from either provers, witnesses and verifiers. Such a secure system benefits to all the users of the system as verifiers will have a high confidence in the location proofs received from users. Moreover, provers and witnesses must be able to control the amount of information disclosed during a verification process.

In this section, we enumerate the threats to be considered while designing each part of a location proof system. Our model differs from the others as we give more possibilities to a malicious verifier in attacking the system. The adversaries we introduced here are the following: *malicious prover*, *malicious witness*, *distance adversary*, *terrorist prover* and *malicious verifier*. We describe each adversary in terms of his capabilities, his goals and the strategies he may use to attain his goal. The adversary can be any user of the system, in particular he may have credentials received from the CA. In this case, they are labelled as malicious because they deviate from the normal running of the protocol. We use the term *intruder* to designate an attacker that is not registered within the system. Intruder can collaborate with malicious users to attack the system. In our security model, we are given the adversary the ability to wiretap communication taking place in his vicinity.

4.3.1 Malicious prover

A malicious prover is registered within the system by the CA. Therefore, he shares the same capabilities as the honest users of the system. More precisely, a malicious prover can communicate with users present in his neighbourhood, he can request location share from them, he can process location shares into corresponding location proofs to convince a verifier of his location claim.

The goal of a malicious prover is to *forge location proofs* using location shares he received from honest witnesses. Forging a location proof means to create location proof using location shares from a different context. To obtain location shares, this adversary acts like in the normal way. More precisely, he run the protocol for the gathering of location shares with witnesses within his proximity. Once the location shares received, the malicious prover tries to create a location proof with a different context than the one included within the location shares.

We say that a location proof system is secure against actions of a malicious prover, if it is impossible for a prover to create location proof with a context different from the context included within the location shares.

4.3.2 Malicious witness

A malicious witness can be any user registered by the CA within the system that deviates from the protocol when he is acting as a witness. Because a malicious witness has valid credential of the system, he has the same capabilities as a witness within the system. He can listen to the network and obtain location proof request from surrounding honest prover and he can create location share for honest prover. The goal of such adversary is to testify a context (location and time) different from the one he received from the honest prover in order to invalidate the future location proof that will be generated by the prover. The malicious witness may also aim at infering the real identity of the prover and to link the different actions of a targeted prover within the system.

4.3.3 Malicious verifier

A malicious verifier is a registered verifier within the system. He is given the authorization from honest prover to process their location proof in exchange of some services. Malicious verifier received location proof from prover. Their objective is to obtain the identity of the witness who have helped the prover in the creation of the proof or the identity of the prover if he is communicating to the verifier over an anonymous channel. By identifying the witness and the time at which the prover has interacted with the witness can help the malicious verifier to infer social network of the user and other valuable information.

4.4 Privacy

Privacy is of central importance to mobile users. Namely, we must prevent issuers and verifiers of location proofs from violating a user's privacy. In the context of ubiquitous computing, privacy of users depends of two major factors that we call *anonymity* and *unlinkability*.

4.4.1 Anonymity

In the domain of location proof systems, anonymity has been defined to require that an adversary, not in possession of the secret information of the CA and not registered in the system, cannot recover the identity of the prover and witnesses from a location proof and the corresponding location shares.

While this notion preserves the indistinguishably of the users interaction according to an attacker external to the system, we need to go beyond to comply with the principle of *data minimization*. Data minimization requires that users share only the minimum set of amount of information to obtain a service. In other terms, users of the system should hold personal data about an individual that is sufficient for the purpose of the service they are providing. We will now see what this means in practice.

We recall that the first principle of a location proof is to allow provers to convince verifiers about their position. The term verifier is very broad as it includes any service to which the prover wants to prove his location. For instance, a verifier may be police authorities to which the prover may want to provide alibi or merchants to which a prover may want to prove their loyalty to obtain discount. Verifier can also be any LBS which requires verification of the location claimed by the prover before granting access to its service. While a prover is willing to share personal information with police authorities, this is not always the case when communicating with LBS. Because of the diversity of entities that can play the role of a verifier, we do not require the verifier to be trusted by the prover.

In this context, applying the data minimization principle shows two supplementary notions of privacy that are *the witness' anonymity with respect to the prover and the final verifier*, and *prover' anonymity with respect to the witnesses and the final verifier.*

In fact, when a prover asks his witness for a location proof, all the witnesses need to know is the context to be certified and nothing more about the prover. Therefore, information like for example the prover's identity public key or any other information that can be used to uniquely identify the prover among the users of the system has to be hidden from the witnesses during the collection of a location share. This requirement may also hold during the verification of location proof in which the verifier is not required to know the identity of the prover nor the identities of the witnesses. The only information that may be available to the verifier is the context of the location proof.

4.4.2 Unlinkability

Unlinkability is a privacy property of crucial importance. Here, a user continuously gathers location proofs. If the (maybe colluding) parties that issue location proofs learn the user's identity, the user would become traceable.

In our context, unlinkability should hold for the witnesses and also for the prover. More precisely, given two gathering sessions S_1 in which the prover P receives a location share σ_{S_1} from a witness W and S_2 in which the prover receives a location share from the same witness. Then any user of the system that has access to the proof σ_{S_1} and σ_{S_2} may not be able to tell if the proof originated the same witness. This is needed here to avoid situation in which attacker can exploit collocation information in order to infer private information about the users.

4.4.3 Location granularity

Granularity designates the level of details considered in a model or decision making process. In our context, location granularity refers to the ability of the system to be flexible in terms of the representation of the location of its users.

Intuitively, a location proof has to contain location information in some form. The location information in the proof not only vouches for the location of the prover, but it might also indirectly reveal sensitive information about that person like his interests to a verifier. For example, if a service is offered only to mobile users in Toulouse, users who present a proof showing that they are at the Airbus site not only prove their qualification to the service, but also reveal more location and personal information than necessary to the service provider. Therefore, the user should be given the ability to control how much location information to disclose in response to the location requirements of different applications and services.

More precisely, the location of a prover may be hidden in the location proof and represented into different levels of granularity, like city, neighbourhood, or an exact geographical point. Therefore, when a prover tries to claim his location to a verifier, he can decide of the level of granularity he is willing to reveal to that particular verifier without revealing more private information than needed.

4.4.4 Location sovereignty

Location sovereignty is the guarantee that users have control over the location shares they have collected.

In fact, in [ZC11], location shares of users are stored on a central server that needs to be contacted by the verifier during the verification of the location claimed by the prover. Storing private information of the users on a remote server poses many privacy concerns. At a high-level, these privacy concerns stem from two issues: first, users must rely on the infrastructure not to be malicious; and second, the infrastructure must provide access control and data sharing policies that are easy to use and satisfy the requirements of users. While both issues are challenging in practice, we suggest to design a system that put users in control of their privacy policies. Therefore, location proof system should put the users in control of the location shares they have collected.

Users can continuously collect location proofs and store them locally on their devices. The role of the infrastructure may be restricted to providing location shares to those provers. Provers can then use the set of location shares they have collected over time for a multitude of services of their choice. This puts users in control to decide how they want to use this information and who they want to share it with.

4.4.5 Location privacy

A location share should protect the location privacy of a prover, by ensuring that the location information does not appear in clear in the proof. Therefore, an attacker cannot deduce the location of the prover from the knowledge of the location share alone. We also require that the location proof does not reveal information about the location of witnesses involved in its generation. Indeed while it is obvious that witnesses are in the communication range of the prover during the creation of the proof, the resulting location share may not reveal more fine-grained location information about witnesses and their identities.

4.5 Security

In this section, we motivate the security properties that are required for location proof systems.

4.5.1 Ownership proof

Location is the basic component in the process of creation of location proof. Prover may collect location shares from immediate witnesses in order to generate corresponding location proof for a particular verifier. A location share represents a piece of contextual information testifying of the presence of the prover nearby the corresponding witness. Location share acts like a testimonial of the witness telling to the verifier that he have seen the prover at position P and time T. Therefore, location proof should include information that can be used to link them to their genuine owners. In other words, location shares and the corresponding location proofs should be personal and not transferable.

4.5.2 Unforgeability

In [WF03], Walters and Felten have defined unforgeability of location proof based on a bipartite approach as being the impossibility for an attacker to create location proof on their own as long as the private keys of the access points are not compromised.

However, in a collaborative approach, user can be both prover and witness depending of the role they are playing during the collection of the proof. Therefore, it becomes difficult for the verifier to know if the proof has been created by the prover himself or by a witness because the prover is allowed to act as his own witness. In this particular case, verifier may require the prover to provide at least two location shares to validate a location proof. With the condition that each witness can deliver only one location share to the prover in a particular context. In practice, it means that it should exists an algorithm which when provided two location shares from the same context, returns true if the shares has been created by the same user and false otherwise. We call this the *uniqueness* of location share.

Therefore, provided that location share has uniqueness property and the prover needs at least two location shares to validate his location proof then the verifier can easily check that the prover can not forge location proofs on his own.

4.5.3 Resistance to localization attacks

Localization attacks constitute a major threat against location proof system. In fact, consider a user situated in a context and trying to obtain location shares from a different context. To succeed this attack, the prover can launch in real-time a wormhole attack against the witnesses situated at the desired location. The wormhole would be responsible for relaying messages between the targeted location and the location of the attacker without the witnesses being able to detect such situations. While many location proof systems do not consider such attacks, some [SW09] has proposed RTT-like techniques echoing challenges between the prover and the witnesses. However, the techniques proposed by such systems are not very effective as they are vulnerable to attacks related to RTT measurement like distance fraud, mafia fraud, terrorist fraud and distance hijacking (See Chapter 5 for more details). To our knowledge, the only secure alternative is the use of distance-bounding protocol.

4.5.4 Collusion prover-prover

A collusion prover-prover [WZP$^+$] puts in relation a malicious prover and one or more colluders (the colluding party). Their objective is to provide the malicious prover with location shares for a context in which the latter is not. In the sequel, we call the context in which the prover is not physically present the *targeted context*.

To be valid, a collusion prover-prover requires the malicious prover to have valid credentials from the CA, with the condition that he never discloses these credentials to the colluding party. The colluder role can be played by either a user registered in the system or not. During the collection of the location shares, the colluder may receive information from the malicious prover in real time. For this purpose, we give the colluder and the dishonest prover a secure communication channel in which they can exchange information.

4.5.5 Collusion prover-witness

The term collusion prover-witness has been first introduced in [WZP$^+$]. It denotes an attack in which a witness colludes with a prover to create a location proof share for the latter, even though one or both of them are not at the location as claimed in the location proof share. To the best of our knowledge, there is no effective solution to detect this type of collusion yet. Thus, it remains one of the most challenging attacks to protect against in location verification.

4.6 Analysis of previous work

Based on the notions we have presented in Section 4.4 and Section 4.5, we investigate the security and privacy properties of state-of-the-art location proof systems. The result of the analysis is shown in the Table 4.6. In terms of notation, a checked cell means that the protocol ensures this property while a blank cell indicates the opposite.

4.7 Conclusion

In this chapter, we have introduced a model for studying the properties, in terms of the security and privacy, offered by location proof systems described in Chapter 3. The adversary model we elaborate differs from the others as we give more possibilities to verifier in attacking the system. In fact, the verifier's role can be played by any party without requiring the prover to trust him. A malicious verifier is interested in learning the identity of the users in order to profile them, thus users need to be anonymous regarding the verifier.

	Protocol Properties	[LH10]	[SW09]	[WF03]	[ZC11]	[GG09]	[TCB12]	[DCF12]	[WZP+]
					APPLAUS	SLVPGP	LINK		STAMP
SECURITY	Correctness	✓	✓	✓	✓	✓	✓	✓	
	Ownership proof	✓	✓	✓	✓	✓	✓	✓	✓
	Unforgeability	✓	✓	✓	✓	✓	✓	✓	✓
	Resistance to distance fraud	✓	✓	✓		✓			✓
	Resistance to mafia fraud		✓			✓			✓
	Resistance to distance hijacking								✓
	No single point of failure	✓	✓	✓				✓	✓
	Resistance to terrorist fraud				✓				✓
	Proof share uniqueness								✓
PRIVACY	Prover anonymity & unlinkability (gathering phase)	✓	✓	✓	✓			✓	✓
	Prover anonymity & unlinkability (verification phase)								
	Witness anonymity & unlinkability (gathering phase)	✓			✓			✓	✓
	Witness anonymity & unlinkability (verification phase)	✓			✓	✓	✓		✓
	Witness location privacy	✓						✓	✓
	Confidentiality	✓				✓			✓
	Location sovereignty	✓							✓

Table 4.1: Comparison of different location proof systems

The comparison shows that many of the location proof systems we analyse are vulnerable to some sort of localization attacks in which malicious prover relays the signal to made them appear at a location where they are not. Therefore, location proof system may provide the witnesses with security mechanisms to properly determine that the prover is present within proximity. The next chapter introduces the concept of distance-bounding protocol, a cryptographic primitive that can be used to threaten the process of gathering location proofs.

Les protocoles démiteurs de distance

L'authentification sur les réseaux conventionnels tels que l'Internet, est construit sur quelque chose que l'utilisateur sait (un mot de passe, etc.), quelque chose que l'utilisateur a (une carte d'accès, etc.) ou ce que l'utilisateur est (biométrie). Dans les réseaux sans fil, l'information de localisation peut être utilisée pour authentifier un périphérique ou un utilisateur. Ce chapitre introduit les protocoles délimteurs de distance ou protocoles de distance-bounding, leurs contextes d'application et les attaques contre lesquelle elles protègent. Les protocoles délimteurs de distance sont des protocoles de sécurité qui permettent à un vérificateur V de s'assurer qu'un prouveur P se trouve à une distance bornée et définie de lui même. Ces protocoles sont issus des travaux réalisés par Stefan Brands et David Chaum en 1993, dont le but est d'améliorer les mécanismes d'authentification traditionnels. Les protocoles délimiteurs de distance permettent entre autre de se prémunir des attaques du type man-in-the-middle (MITM) et de ses dérivés (*cf.* Section 5.3), en calculant la distance séparant V et P. Cette distance est obtenue en multipliant la vitesse de propagation de l'onde électromagnétique par le temps que met P à répondre aux challenges envoyés par V. Il existe deux grandes familles de protocoles délimiteurs de distance. Ceux basés sur la cryptographie symmétrique et ceux basés sur la cryptographie asymmétrique. Les protocoles délimiteurs de distance symmétriques à la différence des protocoles asymmétriques nécessitent que P et V partagent un secret commun au préalable. La sécurité de ces protocoles est évaluée en considérant quatre types d'attaque, que nous décrivons dans ce chapitre ainsi que les modèles de sécurité formelle s'y afférant.

Chapter 5

Distance Bounding protocols

5.1 Introduction

In the previous chapter, we have seen that before sending a location proof to the prover, the witness has to ensure that the prover that has initialized the session is indeed within physical proximity as a countermeasure to wormhole attacks. This chapter introduces distance-bounding (DB): a process whereby a party (the verifier) is assured (i) of the identity of a second party (the prover) and (ii) that the prover is located in his close vicinity (known as neighbourhood).

This chapter starts by defining the notion of distance-bounding before presenting the actors involved and the terminology used in such system (see Section 5.2). Then we investigate the security of DB protocols by presenting the different attacks such protocol must be able to thwart along with the security model used to analyse the security properties of distance-bounding protocol (See Section 5.3). Finally in Section 5.4, we give a security analysis of some existing DB protocols.

5.2 Overview of distance-bounding

Authentication protocols are classically run between two entities namely a prover and a verifier and enable the latter to decide whether the prover is legitimate or not. Such protocols enable access control, and are used in logistics, public transport, or personal identification. However, security models for classical authentication schemes do not capture relay attack in which a Man-In-The-Middle (MITM) adversary just forwards data between the prover and the verifier trying to impersonate an honest prover in front of the verifier. Relay attack is a concept first proposed in 1976 by Conway in a scenario referred to as the *"Grand master chess problem"* [Con76]. In this scenario any player, even a person not familiar with the rules of chess, could play against two grand masters by challenging both of them to a game by post. The player would then simply forward the move received from one grand master to the other, effectively making them play against one another. This results in the player either winning one match, or earning a draw in both matches. The concept of relay attacks as described above has been extended to authentication protocol by Desmedt with an attack on the Fiat-Shamir identification protocol [Des88]. In this scenario, the MITM adversary is typically a coalition of two adversaries, a *leech*, which interacts with the prover, impersonating a verifier, and a *ghost*, which interacts with the verifier, impersonating a prover. The two adversaries usually communicate via fast, reliable communication channels, and by relaying correct, honestly-generated information between them, they ensure that the ghost, which is an illegitimate party, authenticates to the verifier thus defeating the security requirements.

Following the idea that relays seem to cause a processing delay in the MITM attacker, Brands and Chaum introduced Distance-Bounding protocols [BC93] as a feature to enhance traditional authentication mechanisms to withstand MITM relay attacks. In such protocol, a clock is mounted on the verifier, such that it can measure the Time-of-Flight (ToF) between sending a challenge and receiving a response from the prover. The number of challenge-response interactions is determined by a chosen security parameter. To be correctly authenticated, the prover must reply such that the measure RTT of the signal is less than a pre-set value t_{max} representing an upper bound associated with the maximum trusted communication distance. Assuming that the communication speed is constant and very fast, if the protocol succeeds then (1) the verifier is convinced that the prover is legitimate, and (2) the prover is within the maximum distance associated with t_{max}. In a distance-bounding protocol, not all exchanged messages are subject to round-trip-time measurements. In general, a DB protocol can be divided into the three following phases.

1. *Initialization phase*: This is the first step of the protocol in which the prover and verifier exchange information to initialize the session. There is no time-constraint for this phase, which is why it is also called *lazy phase*.

2. *Distance-bounding phase*: During this step, prover and verifier exchange bits at very high speed. The verifier sends a challenge bit to the prover and measures the time t_i from the moment he sent a challenge to the moment the corresponding response is received. In fact, once a challenge bit is received, the prover must compute a response bit using a response function and then forwards the response as soon as possible. The response function serves at the same time to authenticate the prover and to prove that he actually receives the challenges. To effectively obtain a secure distance estimate, the time taken to calculate the response must be minimal and constant. This round is repeated several times to increase the robustness and the security of the estimate provided.

3. *Verification phase*: this step allows the verifier to decide if the authentication succeeds or not.

More formally, by considering a single verifier V and a single prover P, we can define a distance-bounding authentication scheme as follows. .

Definition 1 (Distance-bounding authentication)**.** *A distance-bounding authentication scheme for timing parameters (t_{max}, T_{max}, E_{max}, N_c) is a triplet of efficient algorithms $\mathsf{DB} = (\mathsf{Kg}, \mathsf{P}, \mathsf{V})$ in which:*

(i) *Kg is a key generation algorithm with parameter $n \in N$. Kg generates a keypair sk/pk, the verifier is initialized with the public part of the key, namely pk and the prover with the secret part of the key sk.*

(ii) *The authentication algorithm run on the joint execution of algorithms $\mathsf{P}(sk)$ and $\mathsf{V}(pk)$ generates, depending on t_{max}, T_{max}, E_{max}, N_c, a verifier outputs $b \in \{0, 1\}$.*

The value N_c denotes the number of rounds in the time-critical phase, t_{max} is the upper-bound associated with the timing of each round of the time critical phase. With the value T_{max} represents the number of rounds in the time critical phase exceeding t_{max}. Similarly, E_{max} is the maximum number of time-critical phases with erroneous transmissions. When we have pk=sk , then the DB scheme is said to be symmetric otherwise if $pk \neq sk$, the DB scheme is said to be asymmetric.

We require the *DB* scheme to be complete. Thus, for any pair of key (sk, pk), the decision bit b produced by an honest verifier $\mathsf{V}(pk)$ interacting with honest prover $\mathsf{P}(sk)$ under the requirements following from the timing parameters t_{max}, is 1 with probability (negligibly close to) 1. In addition, the protocol should be sound in the sense that if the decision bit b produced by the honest party $\mathsf{V}(pk)$ is 1, then the party $\mathsf{P}(sk)$ is in the proximity distance as defined by the parameter t_{max}. Such protocols were recently implemented by Rasmussen and co-authors [Rv10].

5.3 Security of distance bounding protocol

The security of distance bounding protocols largely depends on the assumption that the prover's processing time is negligible compared to the measured challenge-response round-trip times. Given that the verifier does not trust the prover and cannot estimate the prover's hardware and processing capabilities, the safest assumption that the verifier can make is that the prover is able to process the challenges and transmit the replies in negligible time. If the verifier overestimates the prover's processing time (*i.e.*, the prover is able to process signals in a shorter time than expected), the prover would be able to pretend to be closer, thus violating the distance bound. Despite this fact, DB protocols must be able to withstand four main types of location attacks that we describe thereafter.

Distance fraud

Definition 2 (Distance fraud.)**.** *In the distance fraud, two parties are involved: one of them (the verifier V) is not aware of the fraud that is going on, the other one (the malicious prover $\bar{P}$ is placed far from the verifier) performs the fraud. The fraud enables $\bar{P}$ to authenticate to V, thus proving a fake statement related to his physical distance to V.*

Distance fraud was first introduced by Brands and Chaum in [BC93]. The fraud is effective when the malicious prover knows the response before he receives the challenge from the verifier, then he can reply in advance, thus fooling the verifier's clock. In practice, the prover can guess the challenge, or he can wisely select some initialization parameters so as to make sure that the responses do not depend of the challenges for some rounds. We say that a protocol is resistant to distance fraud if a dishonest prover situated outside the proximity of the verifier has negligible probability to be accepted.

Mafia fraud

Definition 3 (Mafia fraud.)**.** *In the mafia fraud, three parties are involved: two of them (the honest prover P and the verifier V) are not aware of the fraud that is going, on while the third party (the Man-In-The-Middle adversary consists of two collaborating parties $\hat{V}$ and $\hat{P}$) performs the fraud. $\hat{V}$ impersonates V in front of P and $\hat{P}$ impersonates P in front of V. The fraud enables $\hat{P}$ to convince V of an assertion related to the private key of P.*

This fraud was first introduced by Desmedt in [Des88]. The adversary is modelled by a couple of collaborating entities ($\hat{P}$,$\hat{V}$) relaying messages between P and V. $\hat{P}$ initiates a session with V and $\hat{V}$ initiates a session with V, then they relay messages between the two sessions initiated. The adversary's aim is to be authenticated by the honest verifier. Due to the delay introduced by the transmission of messages between the adversaries ($\hat{P}$,$\hat{V}$), they are not allowed to purely relayed messages during the distance-bounding phase. Purely relay happens when V sends a challenge b to $\hat{P}$, thinking he is communicating with P. $\hat{P}$ sends the received bit to his collaborator $\hat{V}$ before than $\hat{V}$ delivers b to the P. Then, P provides $\hat{V}$ with the corresponding response bit b and then $\hat{V}$ sends the response bit to $\hat{P}$ that sends it to the verifier V. Such kind of interaction is not allowed because due to the delay introduced by the interaction between P and V in the transmission of the challenge and its associated response, this will be detected by the verifier's clock. We say that a distance-bounding protocol is resistant to mafia fraud is the MITM adversary has negligible chance to be authenticated.

Terrorist fraud

Definition 4 (Terrorist fraud.)**.** *In the terrorist fraud, three parties are involved, one of them (the verifier V) is not aware of the fraud going on, the two others (the malicious* *prover P and the adversary A also called terrorist) collaborate to perform the fraud. The help of P enables A to convince V of an assertion related to the private key of P.*

Intuitively, during a terrorist attack, a MiM adversary situated in the vicinity of the verifier actively colludes with a malicious prover (which is situated outside of the proximity of the verifier) to impersonate the latter. While the prover is not allowed to give trivial information such as the whole secret key. The restriction on how much a prover can help the terrorist has been formalized in the distance-bounding literature according to three main security models [ABK+11, DFKO11a, FO13b, Vau13]. The objective here is to discourage malicious provers to collaborate with an adversary because it will implicitly leak helpful information to the adversary to authenticate latter.

The first model [ABK+11] introduced by Avoine and co-authors says that the protocol resists terrorist fraud if when the terrorist authenticates to the verifier then the terrorist gains any information about the secret (even some bit of the secret key). In other terms, to perform such attack the secret key should be information-theoretically hidden during the fraud. This notion is rather weak as many attacks are ruled out such the one proposed by Hancke [Han12] that proposes to use the noise tolerance of DB scheme to perform terrorist attack.

The second notion called GameTF introduced by Fischlin and Onete in [FO13b] stipulates that a protocol resists to terrorist fraud if the terrorist adversary gains advantage to launch better mafia fraud.

The third notion called SimTF introduced by Dürlrhoz and co-authors in [DFKO11a] says that a protocol resists terrorist fraud if there exists a simulator based on the state of the adversary that must always authenticate. Here, the malicious prover can provide the terrorist with any kind of information excluded the data contained in the prover's internal state (the secret key). This definition is very broad, enabling syntactic attacks like the one in [FO12] against the scheme of Reid and co-authors.

As introduced early, the definition of [ABK+11] [ABK+11] is weak but it enables efficient constructions. The notion of SimTF [FO13b] is very strong and difficult to achieve in practice as recognized by its authors. Finally, the GameTF [FO13b], which is a game-based notion, for terrorist-fraud is strong enough for practical applications.

Terrorist fraud is a very strong attack and thus many existing DB protocols (notably including that of Hancke and Kuhn [HK05a]) do not address it. Terrorist Fraud resilient protocols preserve the basic structure of distance bounding protocols, but bind the prover's private key to the nonces that are exchanged in the protocol. This prevents the prover from simply handing over the nonces to the external attacker without disclosing its secret. Most of the terrorist resistant DB protocols are in general symmetric. This is the case for the Swiss-Knife protocol [KAK+09] and for the scheme of Avoine, Lauradoux, and Martin [ALM11], which resists terrorist fraud according to the definition of Avoine and co-authors [ABK+11]. The class of SKI protocols [BMV13b], which thwart terrorist fraud using the definition of [Vau13] in a provable way, is also symmetric.

A notable exception to this approach is the DBPK-Log protocol [BB05a] due to Bussard and Bagga. This protocol also requires secret sharing, but does not involve a shared secret between the prover and the verifier.

Distance hijacking

Another type of fraud, known as *distance hijacking* [CRSC12], has recently been proposed by Cremers, Rasmussen, Schmidt and Capkun. The fraud considers a malicious prover who aims to convince a verifier that he is located within the verifier's neighbourhood. To realize this, he will abuse some other provers who are indeed in the verifier's neighbourhood. For example, a malicious prover can reach his goal by hijacking the fast phase of a distance-bounding protocol executed between an honest (closer) prover and the verifier. Conceptually, distance hijacking is situated somewhere between distance fraud and terrorist fraud. In contrast to terrorist fraud, where a dishonest prover colludes with another attacker, distance hijacking considers a dishonest prover who interacts with (abuses) other honest provers. Unlike distance fraud that only involves a dishonest prover and a verifier, distance hijacking also involves other honest provers. Avoine and Tchamkerten [AT09b] also suggested another attack, namely slow-phase impersonation resistance. This attack is especially applicable to cases where distance-bounding protocols are implemented on resource-constrained devices (*e.g.*, RFID tags) that cannot support many time-critical rounds. Since in many distance-bounding protocols the level of impersonation security heavily depends on the number of time-critical rounds, the overall security level for resource-constrained implementations would then become too low in practice. By employing lazy-phase authentication, the security level of the protocol is increased.

5.4 Overview of the state-of-the-art

As introduced early, Brands and Chaum [BC93] were the first to introduce distance-bounding protocol in the literature. The heart of their protocol resides in the design of the response function. In fact, the response function simply consists of a XOR operation that takes as input a fresh challenge generated by the verifier and a secret information only known to the prover. Then, the prover must reply with the response bit to the verifier as soon as possible to avoid the authentication session to fail. The protocol also uses commitment and signature schemes to intertwine the physical time measurement and the prover authentication. These methods allow the protocol to achieve resistance against distance and mafia frauds with probabitlity $(\frac{1}{2})^m$, with m being the number of rounds during the distance-bounding phase.

Another category of distance-bounding protocol is the one proposed by Hancke and Kuhn [HK05b]. Their idea is to optimize the the Brands and Chaum protocol for RFID devices. Then, their protocol uses precomputed registers to compute the response bits during the distance-bounding-phase and no additional messages need to be transmitted during the verification stage. However, with the computation lightening, Hancke and Kuhn's protocol has two issues: its vulnerability against the terrorist fraud, and its non-optimality concerning mafia and distance frauds (*i.e.,* the protocol optimal bound against mafia fraud is $(\frac{3}{4})^m$). All Hancke and Kuhn's protocol descendants attempt to solve one of these issues.

Bussard and Bagga had the idea to interleave the prover's secret and his answers during the fast phase to prevent terrorist fraud. Their protocol [BB05a] uses the precomputed registers due to Hancke and Kuhn, but it has mafia fraud resistance to $(\frac{1}{2})^m$ at the cost of a complex zero-knowledge proof of knowledge. To decrease the computational cost of the Bussard and Bagga protocol, Reid and co-authors introduce a terrorist resistant distance-bounding protocol [RNTS07] with a mafia security bound of $(\frac{3}{4})^m$. Its direct descendant, Tu and Piramuthu's protocol [TP07] proposes a protocol composed of a succession of fast and slow phases. However this protocol suffers from vulnerabilities, and two articles [KAK$^+$09, MP08] attacked it. These attacks demonstrate in particular that the prover's secret can be leaked to an eavesdropper. Moreover, this attack can be fasten, if the adversary interacts with the parties during the protocol session. The other descendant of the protocol of Reid and co-authors, the Swiss-knife [KAK$^+$09] fixes the poor mafia fraud resistance problem by adding a third phase. In addition, it provides mutual authentication. In [PLHCvdLT09], the authors claim that they found an attack on the Swiss-knife, and propose Hitomi, a descendant of the latter, to solve this issue. However, the attack is based on nonces repetitions. Moreover, under the same assumption, nonces repetitions, Hitomi suffers, to a similar flaw. The other Swiss-knife descendant [ALM11] explicitly introduces secret-sharing to counter the terrorist fraud, and studied the best settings to use it. The study led to the explanation of some vulnerabilities found in previous protocols designed to mitigate the terrorist fraud. Table 5.1 depicts a comparison of the properties of several well-known distance-bounding protocols.

5.5 Conclusion

This chapter has introduced the concept of distance-bouding protocol, which is a cryptographically secure protocol aiming to prove the proximity of two devices relative to each other. We have also seen that the security of these protocols is evaluated against five mains attacks which are the *distance fraud*, *mafia fraud*, *terrorsit fraud*, *impersonation* and *distance hijacking*. The resistance to these attacks has been covered in literature within three main security models: namely DFKO [DFKO11a], and ABKLM [ABK$^+$11] and [Vau13]. The models cover resistance to distance fraud, mafia fraud, terrorist fraud and impersonation in respectively formal and informal methods. Actually, the ABKLM is considered outdated because it has been shown by Hancke that its definition of terrorist fraud resistance is rather weak (a terrorist adversary can still defeat the protocol by exploiting the noise tolerance threshold). Many of the distance-bounding protocol offering resistance

	Fraud Protocol	Impersonation	Mafia	Terrorist	Distance
Symmetric	Reid et al. [RNTS07]	1	?*	No	?*
	Tu & Piramuthu [TP07]	1	1	No	$\left(\frac{3}{4}\right)^m$
	Swiss-Knife [KAK$^+$09]	$\left(\frac{1}{2}\right)^m + \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Unf}}$	$\left(\frac{1}{2}\right)^m$	Yes	$\left(\frac{3}{4}\right)^m$
	Avoine and co-authors [ALM11]	1	$\left(\frac{2}{3}\right)^m$	Yes	?**
	$SKI_{\mathbf{pro}}$ [BMV13a]	$(\frac{1}{2})^m$	$\left(\frac{2}{3}\right)^m$	Yes	$\left(\frac{3}{4}\right)^m$
Asymmetric	Brands and Chaum [BC93]	$\left(\frac{1}{2}\right)^m + \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Unf}}$	$\left(\frac{1}{2}\right)^m + \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Unf}}$	No	$\left(\frac{1}{2}\right)^m$
	GOR [GOR14]	$2^{-\lvert\mathbb{G}\rvert}$	$\left(\frac{1}{2}\right)^m$	No	$\left(\frac{3}{4}\right)^m$
	DBPK-Log [BB05a]	$\left(\frac{1}{2}\right)^{4m'}$	$\left(\frac{1}{2}\right)^m$	No	1

Table 5.1: Success probability of classical attacks against well-known DB in noiseless conditions. * No generic bounds applicable; one specific instantiation is NOT Mafia-Fraud resistant, but resists distance fraud with probability $\frac{3}{4}$ per round [FO13a]. ** Bay and co-authors show that stronger assumptions on f are necessary to achieve the claimed bound of $\frac{3}{4}$ per round. The value $m' := 50$ is a system parameter defined by [BB05a].

to all types of frauds found in the literature are symmetric. However symmetry is not a desirable property to ensure privacy within location proof system. Furthermore, the only terrorist resistant distance-bounding protocol that offer resistance to terrorist fraud has been showed insecure by Reid and co-authors. Therefore, the next chapter present our proposal for an asymmetric distance-bounding protocol which offer provable security against all the frauds using the DFKO model.

VSSDB: Un protocol délimiteur de distance asymmétrique

Dans ce chapitre, nous introduisons la première contribution de nos travaux présentés dans ce manuscript. Il s'agit de VSSDB (*Verifiable Secret Sharing Distance-Bounding protocol*), un protocol délimiteur de distance basé sur la cryptographie asymmétrique. Dans ce chaptire, nous commençons par décrire DBPK-Log, un protocole délimiteur de distance asymmétrique proposé par Laurent Bussard et Walid Bagga. DBPK-Log est basé sur la cryptographie asymmétrique et a été initialement conçu pour résister à l'attaque terroriste. Cependant, Bay *et al.* ont récemment montré des limites quant à la sécurité de DBPK-Log contre les attaques terroristes et par distance.

À la différence des protocoles délimiteur de distance présents dans la littérature, VSSDB utilise la version vérifiable des schémas de partage de secret. Ainsi, à partir d'une clé publique, le vérifieur est capable d'authentifier bit par bit la clé secrète correspondante, empêchant ainsi le scénario d'attaque terrosite décrit par Bay *et al.* contre DBPK-Log. De plus, la fonction de réponse de VSSDB est basé un système de *modes*. Les modes sont des bits choisis de manière aléatoires par le vérifieur lors de la phase d'initialisation du protocole. Ils sont utilisés en complément des challenges habituels émis par le vérifieur lors de la phase d'échange rapide. Nous prouvons par la suite qu'ils permettent effectivement d'améliorer la sécurité générale du schéma proposé. VSSDB est construit sur une fonction de réponse pouvant utiliser au choix deux ou quatre modes de réponses. D'un point de vue pratique, opter pour la fonction à deux modes implique une réduction des temps de calcul mais nécessite que les clés privées soient générées par une autorité centrale et de confiance; Ce qui n'est pas le cas pour la fonction à quatre modes.

Finalement, nous prouvons la sécurité formelle de notre protocole contre la fraude distance, la fraude mafia et la fraude terroriste.

Chapter 6

VSSDB: an asymmetric distance-bounding protocol resistant to terrorist fraud

6.1 Introduction

Relay attacks are a critical threat to both authentication and proximity-testing protocols, as demonstrated for RFID-based Passive Keyless Entry and Start (PKES) systems in cars [FDv11], NFC smartcards [FHMM10], geosocial networks such as Foursquare [CP12] and location proof systems. DB protocols (see Chapter 5) were specifically designed by Brands and Chaum [BC93] to counter this type of attacks. DB specifically enables a verifier to authenticate the prover only if the latter is within a specific proximity and knows a secret key registered within the system. Indeed, the DB verifier is equipped with a clock used to measure the RTT between the sending of a challenge and the reception of the associated response from the prover. The measured RTT is compared to a predefined bound t_{max}, which corresponds to a trusted distance (also called proximity) to the verifier. DB protocols may aim to prevent several attacks, amongst which terrorist fraud, where a dishonest prover helps the adversary to authenticate, but without passing data that allows the adversary to later authenticate on its own. However, most of the DB scheme that offer resistance to this type of fraud are symmetrics then a verifier can always link the actions of a prover within the system. Therefore they do not fit the privacy requirements for location proof systems.

In this chapter, we introduce a new distance-bounding protocol called VSSDB (A Verifiable Secret Sharing Distance-Bounding protocol) for use within location proof system. Like the original protocol of Bussard and Bagga [BB05a], VSSDB is asymmetric and uses homomorphic bit commitment to hide the secret of the prover within the nonces exchanged during the session. In addition, the prover can demonstrate to the verifier that he knows the secret key used to create the nonces used during the session without revealing it. This chapter is organized as follow: in Section 6.2, we review the basis of the Bussard and Bagga protocol (namely DBPK-Log) and why it fails to achieve resistance to terrorist and distance frauds. In Section 6.3, we present the concept of verifiable secret sharing and how it can be used to implement distance-bounding through the presentation of toy protocol. Then, we propose our practical solution for a secure and asymmetric distance-bounding protocol (cf. Section 6.4). Afterward, in Section 6.5 we prove the security of our protocol in the formal security model proposed by Dülrhoz, Fichlin, Kasper and Onete (DFKO) in [DFKO11b]. Finally, Section 6.6 discusses some practical considerations about the protocol and Section 6.7 summarizes the present chapter.

6.2 The distance-bounding proof of knowledge protocol

In general, most DB protocols in the literature that offer resistance to terrorist fraud are symmetric (*e.g.*, they use a secret shared between the prover and the verifier). This is the case for the Swiss-Knife protocol [KAK$^+$09] and for the scheme of Avoine and co-authors [ALM11], which address terrorist fraud by using secret sharing. In these schemes, the shared private key is masked by another random string and used to respond to fast challenges. The class of SKI protocols [BMV13b], which prevent terrorist fraud in a provable way, is also symmetric. Asymmetric distance-bounding protocols were proposed by Gambs and co-authors [GOR14], Hermans and co-authors [HPO13] and Brands and Chaum [BC93], but they fail to thwart terrorist fraud. Bussard and Bagga were the first to consider terrorist-fraud resistance in an asymmetric setting [BB05a]. The next section introduces their protocol and shows the most recent attacks that have been developed against it in [BBM$^+$12].

6.2.1 Presentation

Bussard and Bagga have proposed DBPK-Log, which is a distance-bounding protocol based on a proof of knowledge and a commitment scheme. This protocol (see Figure 1) is run in a cyclic multiplicative group $\mathbb{G}$ of prime order p −1 generated by an element g. The prover $\mathcal{P}$ has a secret key x along with an associated public key $y := \Gamma(x) = g^x$ certified by a trusted third party and known to the verifier $\mathcal{V}$. DBPK-Log is based on three ingredients: a $(2, 2)$ secret-sharing scheme (*i.e.*, two shares are distributed and both are needed to reconstruct the secret), a homomorphic bit commitment scheme (namely Pedersen's commitment [Ped92]) and a zero-knowledge proof of knowledge (ZKPoK) protocol.

During the *initialization phase* the prover picks a random m-bit integer k and uses it to encrypt the long-term secret x as $e \leftarrow \mathcal{E}_k(x) = x - k \bmod p$. The values k and e act as shares in a (2,2)-secret-sharing scheme. Then, the prover commits in a bitwise

Protocol 1: DBPK-Log [BB05a]

Verifier V		**Prover** P
Public key $y := \Gamma(x)$		Private key x
	INITIALIZATION PHASE	
		$k \xleftarrow{\$} \{0,1\}^m$ $e \leftarrow \mathcal{E}_k(x)$ **For** $i = 1 \cdots m$ $v_i, w_i \xleftarrow{\$} \{0,1\}^*$ $a_i = \Omega(k_i, v_i)$ $b_i = \Omega(e_i, w_i)$
	$\xleftarrow{\{a_i, b_i\}_{i=1}^m}$	
	INTERACTIVE PHASE **For** $i = 1$ to m	
Choose $c_i \xleftarrow{\$} \{0,1\}$	$\xrightarrow{c_i}$	$r_i = \begin{cases} k_i & \text{if } c_i = 0 \\ e_i & \text{otherwise} \end{cases}$
Measure δt_i	$\xleftarrow{r_i}$	
	VERIFICATION PHASE	
	$\xleftarrow{\gamma_i}$	$\gamma_i = \begin{cases} v_i & \text{if } c_i = 0 \\ w_i & \text{otherwise} \end{cases}$
	$\xleftrightarrow{\mathsf{ZKPoK[well\ formed]}}$	

Accept if and only if all r_i are coherent with respect to a_i, b_i, γ_i, $\delta t_i \leq t_{\mathsf{max}}$ $\forall i$, and ZKPoK verifies.

manner to each bit of each share using a homomorphic commitment scheme[1], which leads to commitments of the form $a = \{a_i\}_{i=1}^m$ and $b = \{b_i\}_{i=1}^m$, with $a_i = \Omega(k_i, v_i) = g^{k_i} h^{v_i} \bmod p$ and $b_i = \Omega(e_i, w_i) = g^{e_i} h^{w_i} \bmod p$. Finally, this string of commitments is sent to the verifier, thus completing the initialization phase.

Afterwards during the distance-bounding phase, the prover and verifier exchange binary challenges and their corresponding binary responses to estimate the RTT. Each response is equal to either a bit of the random share k or a bit of the encrypted secret e.

Finally during the *verification phase*, the prover sends the randomness (either v_i or w_i) used to generate the commitment of the fast round responses (this value is denoted as γ_i). Since the commitment is homomorphic, the verifier can compute $z = \prod_{i=1}^m (a_i b_i)^{2^{i-1}} = \Omega((k+e), v)$. Then, the prover and verifier runs a zero-knowledge proof of knowledge (ZKPoK) protocol in which the prover demonstrates the knowledge of a tuple (x, v) such that $z = \Omega(x, v)$ and $y = \Gamma(x)]$. The verifier accepts if (1) all the fast responses are

[1] The authors suggest to use Pedersen's commitments [Ped92].

coherent with respect to the committed values, (2) all RTT values are below t_{max} and (3) the ZKPoK succeeds. .

6.2.2 Attacks against DBPK-Log

Avoine, Lauradoux and Martin [ALM11] proposed a key-recovery attack against DBPK-Log as well as other DB protocols. This key-leakage attack also enables to conduct a mafia fraud against another protocol due to Bussard and Bagga (see [FO13a] for more details). In this attack, the adversary uses faulty challenges during the protocol to get the two shares and thus recover the key. Note, However, this key-recovery attack does not work if the prover is honest in the DBPK-Log protocol. Indeed when sent the complement of the challenge, the prover will also send the randomness to open the commitment for the complementary challenge, thus preventing the adversary from knowing whether or not $k_i = e_i$ for some round i. They conclude that a $(t, 3)$ secret-sharing scheme should be used, with $t \geq 3$ an arbitrary number of shares, instead of the original $(2, 2)$ scheme.

Recently, at Inscrypt 2012, Bay and co-authors [BBM$^+$12] proposed another terrorist fraud whose probability of success is $\frac{1}{2}$ against DBPK-Log. Their attack is based on the observation that the homomorphic commitment check proceeds over *all* the commitments. Thus, the adversary $\mathcal{A}$ can generate random values for the shares k and e and use them, as long as it sacrifices one of the commitments to add a value given by the prover ensuring the correct homomorphic verification. More precisely, the prover computes $z' := \Omega(x, v')$ with a newly generated v' and sends z' to the adversary $\mathcal{A}$. Now $\mathcal{A}$ generates two random m-bit strings k and e, for which he generates honestly the commitments a_i and b_i (for $i = 1$ to m). Using these values, $\mathcal{A}$ constructs two strings of commitments to send to the verifier, using as many of the honestly-computed commitments as possible, but also ensuring that the homomorphic verification still holds. More specifically $\mathcal{A}$ sends two arrays A and B with m elements each, which he fills from position 2 to m with the commitments a_i and b_i that he has generated himself. For A_1 and B_1, the adversary guesses c_1 with probability $\frac{1}{2}$ and inserts in the corresponding array (A if $c_i = 0$ and B otherwise) the correct, self-generated commitment (*i.e.*, a_1, respectively b_i). In the other array (corresponding to the complement of c_1), $\mathcal{A}$ adds a value ensuring that $z = \prod_{i=1}^{m} (A_i B_i)^{2^{i-1}} \bmod p$. Now $\mathcal{A}$ can correctly answer to all challenges and reveal unconditionally the correct randomness for the rounds 2 to m as well as the correct randomness for the round 1 if he has correctly guessed c_1. Afterwards, the malicious prover helps the adversary to succeed in conducting the ZKPoK. Thus, no information about x is leaked (except a commitment and a ZKPoK, which reveal nothing) and the adversary $\mathcal{A}$ wins with probability $\frac{1}{2}$.

6.3 Verifiable secret sharing

In cryptography, secret sharing refers to a method that allows one entity called the *dealer* to distribute a secret amongst a group of n users. In fact, the dealer gives different share of the secret to the users, but only when specific conditions are fulfilled will the users be able to reconstruct the secret from their shares. For example, the secret can be reconstructed only when a sufficient number of shares are combined together. Secret sharing has been introduced independently by Adi Shamir [Sha79] and George Blakley [B$^+$] in 1979. The idea to use secret-sharing within distance-bounding protocol is due to Avoine and co-authors [ALM11]. Indeed, they proved that secret sharing can counter terrorist fraud, and detail a method that can be applied directly to most existing distance bounding protocols to implement resistance to such attack in the framework [ABK$^+$11]. However, their framework only works for DB protocols in which the prover and the verifier share the same secret key (symmetric DB). Therefore, the prover can not cheat while creating the nonces (which are the shares of his secret key) because this will be detected by the verifier. However, when the distance-bounding protocol is ran in a asymmetric setting, we propose to switch to another variant of secret sharing called *Verifiable Secret Sharing* (VSS).

A secret sharing scheme is verifiable if auxiliary information is included that allows the users to verify their shares as consistent. More formally, verifiable secret sharing ensures that even if the dealer is malicious there is a well-defined secret that the players can later reconstruct. The concept of verifiable secret sharing (VSS) was first introduced in 1985 by Chor and co-authors [CGMA85].

A VSS protocol consists of two phases: a sharing phase and a reconstruction phase.

1. *Sharing*: initially the dealer holds a secret as input and each user holds an independent random input. The sharing phase may consist of several rounds. At each round each player can privately send messages to other players and it can also broadcast a message. Each message sent or broadcasted by a user is determined by his input, his randomness and messages received from other players in previous rounds.

2. *Reconstruction*: in this phase each player provides his entire view from the sharing phase and a reconstruction function is applied and is taken as the protocol's output.

Verifiable secret sharing is important for secure multi-party computation [Gol98]. Multi-party computation is typically accomplished by making secret shares of the inputs, and manipulating the shares to compute some function.

In the following we present a simple VSS scheme that is the protocol by Paul Feldman in [Fel87b]. The scheme is based on Shamir's secret sharing scheme combined with any homomorphic encryption scheme. Then, we show how this algorithm can be used to implement distance-bounding protocol.

6.3.1 Feldman's verifiable secret sharing

Feldman's verifiable secret sharing [Fel87a] combines a classical secret sharing scheme and an homomorphic commitment scheme. Let us consider g the generator of a cyclic group G of prime order p. The dealer wants to distribute a secret x over n participants such that each group of t participants can reconstruct the secret. The dealer generates (and keeps secret) a random polynomial

$$P(X) = x + \alpha_1 \times X + \cdots + \alpha_t \times X^t.$$

The coefficients α_i are chosen in $\mathbb{Z}_p$. The dealer also computes $\beta_i = g^{\alpha_i}$ and $y = g^x$. Finally, he reveals publicly all the β_i as well as y, g and p. Each of the n participants receives as input a share $s_i = P(i)$, which can be verified using the following formula:

$$g^{s_i} = y \times \prod_{j=1}^{t} (\beta_j)^{i^{j-1}}.$$

6.3.2 Application of verifiable secret-sharing to distance-bounding

Protocol 2 shows how verifiable secret sharing can be used to create a DB protocol. This toy protocol has a unique round compared to regular distance-bounding protocol and is described here only for pedagogical purpose. In this example, we use a $(3, 3)$ secret sharing scheme, which means that all the three shares are needed to recover the secret.

Protocol 2: Toy protocol

Verifier V		**Prover** P
Public key $y = g^x$		Private key x
	Initialization phase	
		$k, \ell \in_R G$ $e = x - k - \ell$ $a = g^k$, $b = g^\ell$ $d = g^e$
	$\xleftarrow{a,b,d}$	
	Interactive phase	
Choose $c \in_R \{0, 1, 2\}$	$\xrightarrow{c}$	$r = \begin{cases} k & \text{if } c = 0 \\ \ell & \text{if } c = 1 \\ e & \text{otherwise} \end{cases}$
Measures δt	$\xleftarrow{r}$	
	Verification phase	

Accept if and only if $y =$all are coherent with respect to a_i, b_i, γ_i, $\delta t_i \leq t_{\mathsf{max}}$ $\forall i$, and ZKPoK verifies

Initialization phase. The prover generates randomly two shares k and ℓ and computes $e = x - k - \ell$. He also computes $a = g^k$, $b = g^\ell$ and $d = g^e$, which are respectively the commitments of k, ℓ and d. Then, he sends these commitments to the verifier.

Interactive phase. During the interactive phase, the verifier and the prover use a single challenge/response to measure the time of flight. Depending on the value of c, the prover answers with a given share.

Verification phase. The verification phase is composed of two steps. First for each share, the verifier checks that the received share matches the commitment and that the timing respects a predefined upper bound t_{max}. Then, he verifies that all the shares are related to y. For instance, consider the case that $c = 0$, which corresponds to $r = k$. In this situation, the verifier checks that the relation $y = g^r \times b \times d$ holds.

Sketch of the security analysis. We do not give here a complete security analysis of our toy protocol. Nonetheless, we briefly discuss the resistance of this protocol with respect to an impersonation attack. Let us assume that an adversary knows y and executes the protocol with a verifier. The adversary picks his own k' and ℓ' and computes $a' = g^{k'}$, $b' = g^{\ell'}$ and $d' = y \times g^{-k'} \times g^{-\ell'}$. However, even if he cannot compute $e' = x - k' - \ell'$, we still have $y = a' \times b' \times d'$. Then, the adversary sent a', b' and d' to the verifier. During the interactive phase, the adversary answers randomly if $c = 2$. All the values used during the verification are consistent except if $c = 2$. In this case, the probability that he gave the correct share is $\frac{1}{|G|}$, in which $|G|$ is the size of the group. Therefore, the success probability of an impersonation attack is:

$$\mathbf{P}(\text{impersonation}) = \frac{2}{3} + \frac{1}{3|G|}. \tag{6.1}$$

One possible modification to enhance the security of this toy protocol could be to chose a different secret sharing scheme such as the following one.

$$\begin{aligned} e &= x - k - \ell, \\ f &= x + k, \\ h &= \ell - x. \end{aligned}$$

In this version, the prover computes $a = g^f$, $b = g^h$ and $d = g^e$. For the verifier, this modification has no consequence and the protocol is unchanged from his point of view.

From the knowledge of y, an adversary can compute valid commitments but cannot only answer randomly when a share is asked. However, this is not the best strategy for the adversary. For instance, suppose that the adversary still picks his own k' and ℓ' and computes $a' = g^{k'}$, $b' = g^{\ell'}$ and $d' = y \times g^{-k'} \times g^{-\ell'}$. While he cannot compute $e' = x - k' - \ell'$, it is still true that $y = a' \times b' \times d'$. Thus, his exact probability of success is still described by Equation (6.1). The choice of the secret-sharing scheme has no impact on the security level because of the homomorphic commitment.

In this toy example, the prover and the verifier have exchanged values belonging to large group during the interactive phase, while in most DB protocols it is generally assume that single bits are exchanged during this phase. In the following, we propose a protocol based on verifiable secret-sharing that achieves stronger security properties.

6.4 Verifiable secret-sharing based distance-bounding protocol

Our construction, called VSSDB, is summarized in Protocol 3. In this section, we first discuss the intuition behind it before describing it in details. In addition, we define a relaxation of the GameTF terrorist-fraud resistance [FO13b]. We also prove that VSSDB achieves this relaxation, called KeyTF security. Finally, we provide in 6.5.5 a version of VSSDB achieving GameTF-security, while preserving the resistance to other frauds of the original VSSDB protocol.

6.4.1 Overview of the protocol

The main idea behind our protocol is to make the prover choose at each round of the distance-bounding phase two m-bit strings, denoted respectively k_i and ℓ_i, that are used to hide the bit x_i of the private key x among three values k_i, ℓ_i, and e_i. At initialization, the prover generates these values and sends them to the verifier. The latter generates a m-bit random value, essentially a string of m-bit values $\mathsf{M}_i \in \{0, 1\}$, encoding a specific *response mode* among two possible ones. The response bits subsequently given by the prover during the time-critical rounds will depend both on the fast-phase challenges *and* on the mode. In particular, the response function, denoted f, takes as input M_i and the round challenge c_i, and outputs a corresponding response r_i. The fact that the modes are chosen honestly by the verifier and the response components (*i.e.*, k_i, ℓ_i and e_i) are committed before receiving the mode values enforce distance-fraud resistance. We hide the committed values from the adversary by encrypting them into Enc with the verifier's public key. Furthermore, we prevent the adversary from forwarding self-generated values by including a signature with the prover's private key on these values within the plaintext to be encrypted. Finally, we avoid replay attacks by making the verifier send a session-specific nonce at each session. The use of homomorphic commitments allows one to verify that $a_i \times b_i \times d_i$ corresponds to the i-th component of the prover's public key denoted as Com_i. Furthermore, a non-interactive zero-knowledge (ZKPoK) proof of knowledge ensures that the value used by the prover is really his secret key, consistent with the values Com_i and with the ciphertext Enc.

During the time-critical rounds, the verifier measures and stores the RTT of each exchange, denoted δt_i. In order to enable the verifier to check the response values r_i, the prover must then open the relevant commitments of the values used to generate the response. Thus, in a subsequent verification step, the prover sends the necessary auxiliary information (the randomness used to commit and possibly the response values themselves), which we denote as a function γ, which takes two inputs M_i and c_i. Due to the use of commitments, the values of x_i are hidden in each session of the protocol. Finally, we use the countermeasure proposed in [KAK$^+$09] and sign the session transcript (*e.g.*, the modes, challenges, and responses used in this session, as well as the session-specific nonce) thus ensuring that a MIM adversary cannot tamper with the exchanged messages and their order.

More specifically, the protocol consists of the following phases.

6.4.2 Setup phase

The Trusted Third Party (TTP) begins by setting-up the global parameters: he picks at random two distinct large prime numbers p and q and computes their product $n \leftarrow pq$. The TTP also generates $t \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ such that $t \neq \pm 1$ of order $\varphi(n)/4$ and computes $s = t^2 \mod n$. In particular, it holds that $s^2 = 1 \mod n$ and, for any two bits a and b, it holds that $s^{a+b} = s^{a \oplus b}$. The parameters n and s are considered to be public and thus known to all parties, while p and q are secret parameters known only to the TTP. These parameters will enable the use of the (homomorphic) commitment scheme (originally called a blob) due to Brassard, Chaum and Crépeau [BCC88].

6.4.3 Registration phase

All the provers and verifiers are initialized by the TTP acting as a registration authority. At prover registration, the TTP generates a private/public signature keypair:

$$(\mathsf{sk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Sign}}) \leftarrow \mathsf{SSKGen}().$$

The signature scheme is modelled as a tuple $\mathcal{S} = (\mathsf{SSKGen}, \mathsf{Sign}, \mathsf{Vf})$ and can be any unforgeable signature scheme (*e.g.*, El-gamal or DSA). The TTP also generates a secret x for the prover. Afterwards, each bit x_i is committed to a value Com_i as follows: the TTP generates a witness z_i and sets

$$\mathsf{Com}_i \leftarrow \mathsf{z}_i^2 s^{x_i} \mod n \,; \mathsf{z}_i = \mathsf{H}^i(x).$$

In this equation, H denotes a cryptographically secure hash function modeled as a random oracle, while $\mathsf{H}^i(x)$ represents the i^{th} iteration of this function on input x. The tuple $\mathsf{Com} := (\mathsf{Com}_1, \ldots, \mathsf{Com}_m), \mathsf{pk}_{\mathsf{Sign}}$ is the prover's public key and is given to all verifiers. We delegate the generation of the keypair $(\mathsf{sk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Sign}})$ and x to the TTP to rule out some attack using some weak key (see Section 6.6 for more details).

6.4.4 Initialization phase

At the beginning of the protocol, the verifier generates a session-specific nonce NV that he sends to the prover. We add at the beginning a 0 to this value as an encoding, to differentiate between an honest protocol run and the *all-or-nothing* disclosure function. As intuitively described above, at the beginning of each session the prover must generate two m-long bitstrings k and ℓ. These strings are used to symmetrically encrypt (we specifically use bitwise XOR) each bit x_i of the private key.

$$e_i \leftarrow \mathcal{E}_{k_i||\ell_i}(x_i) = x_i \oplus k_i \oplus \ell_i.$$

Subsequently, the prover uses a secure bit-commitment scheme $\mathsf{Cmt} = (\mathsf{Commit}, \mathsf{COpen})$ to commit to each bit k_i, ℓ_i, and e_i, using respectively the randomly generated witnesses u_i, v_i, and a value $w_i \leftarrow (u_i v_i)^{-1}\mathsf{H}^i(x) \mod n$. Each commitment takes two inputs, the committed value b and the randomness r, and outputs $\mathsf{Commit}(b, r) := r^2 s^b$ for the parameter s generated at setup.

$$\begin{aligned} a_i = &\quad \mathsf{Commit}(k_i, u_i) = {u_i}^2 \times s^{k_i} && \mod n \\ b_i = &\quad \mathsf{Commit}(\ell_i, v_i) = {v_i}^2 \times s^{\ell_i} && \mod n \\ d_i = &\quad \mathsf{Commit}(e_i, w_i) = {w_i}^2 \times s^{e_i} && \mod n \end{aligned}$$

Afterwards, the prover sends all the committed values a_i, b_i, and d_i to the verifier, who generates randomly a sequence of m response modes $\mathsf{M}_i \in \{0, 1\}$. The modes $\mathsf{M} \in \{0, 1\}$ form one of the two inputs required for the computation of the responses during the time-critical rounds. Finally, the prover pre-computes the responses for each received mode M_i and for each possible bit challenge $c_i = \{0, 1\}$. The responses function $\mathsf{f} : \{0, 1\} \times \{0, 1\} \longrightarrow \{0, 1\}$ is computed as follows, in which for a bit b we use the notation $\bar{b}$ to denote the complementary bit (*i.e.*, $\bar{b} := b \oplus 1$):

$$\mathsf{f}(\mathsf{M}_i = 0, c_i) = \begin{cases} e_i & \text{if } c_i = 0 \\ k_i \oplus \ell_i & \text{if } c_i = 1 \end{cases} \qquad \mathsf{f}(\mathsf{M}_i = 1, c_i) = \begin{cases} k_i & \text{if } c_i = 0 \\ \overline{e_i \oplus \ell_i} & \text{if } c_i = 1 \end{cases}$$

Furthermore, for each tuple of inputs (M_i, c_i), the function $\gamma : \{0, 1\} \times \{0, 1\} \longrightarrow \mathbb{Z}^{2|u_1|} \times \mathbb{Z}^{2|a_1|}$ lists values allowing the verifier to later open the relevant commitments amongst a_i, b_i and d_i and verify the responses. Although we define the range of γ to be a tuple of four values, two of the bit-length of the randomness to commit (*i.e.*, u_i) and two of the bit-length of the output commitments (*i.e.*, a_i), the prover sometimes needs to send only a *subset* of these values. In this case by convention the remaining output values are ignored by the verifier. More precisely for each input (M_i, c_i), the function γ outputs the values of the shares used to compute $\mathsf{f}(\mathsf{M}_i, c_i)$ if these are not given in clear (but rather XORed) and the relevant witnesses. For example, since $\mathsf{f}(0, 0) = e_i$, the corresponding

$\gamma(0,0)$ is equal to w_i, which is the randomness used to construct the commitment d_i of e_i. In contrast, $\mathsf{f}(0,1) = k_i \oplus \ell_i$, thus the output $\gamma(0,1)$ will consist of the two component shares k_i, ℓ_i, as well as the randomness u_i, v_i that opens the respective commitments a_i, b_i. The complete output space of γ is given in Figure 3.

6.4.5 Distance-bounding phase

The prover and verifier then begin the time-critical rounds, which are effectively a sequence of m challenge-response exchanges in which the verifier sends a bit challenge $c_i \in \{0,1\}$ and the prover replies with the response bit $\mathsf{f}(\mathsf{M}_i, c_i)$. The verifier stores δt_i the RTT of the exchange for each round. We point out that implementing a precise measurement of the RTT during the challenge-response rounds is still a difficult technological challenge, although one implementation has been recently proposed [RTv+12].

6.4.6 Verification

During this phase, the prover provides the output $\gamma(\mathsf{M}_i, c_i)$ for each round, as well as a final signature over the entire transcript of the protocol generated with his signature key $\mathsf{sk}_{\mathsf{Sign}}$. Then, the verifier performs the following verification steps.

1. He checks the received signature $\hat{\sigma}$ based on his view of the transcript,
2. he opens the commitments and validates the received responses,
3. he verifies that $\delta t_i \leq t_{\mathsf{max}}$ for all $i = 1, \ldots, m$, in which t_{max} is the trusted proximity bound,
4. he checks that for each $i = 1, \ldots, m$, it holds that $\mathsf{Com}_i = a_i b_i d_i \mod n$.

If any of these verification steps fails, the verifier aborts, which means rejecting the prover. Otherwise, the verifier accepts (*i.e.*, authenticates) the prover.

6.5 Security analysis

In this section, we present the security proofs for the Vssdb protocol from Section 6.4, providing in particular exact bounds for its security. More precisely, our protocol is analyzed in the DFKO security model [DFKO11a] in which terrorist fraud is defined using a game based approach called GameTF.

6.5.1 Resistance against distance fraud

Theorem 1 (Distance fraud resistance). *For any (t, q_V)-distance-fraud adversary $\mathcal{A}$ against VSSDB with m challenge-response exchanges, there exists an adversary $\mathcal{A}_1$ against the* *binding property of the commitment scheme* Cmt *such that*

$$\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{Dist}} \leq 4mq_V \boldsymbol{Adv}_{\mathcal{A}_1}^{\mathsf{Cmt.Bind}} + \left(\frac{3}{4}\right)^m$$

Proof. Our proof relies on the fact that, since for each mode $\hat{\mathsf{M}}_i$, the XOR sum of the responses for the two challenges $\mathsf{f}(\hat{\mathsf{M}}_i, 0) \oplus \mathsf{f}(\hat{\mathsf{M}}_i, 1) = x_i$, a value that is 1 with probability exactly $\frac{1}{2}$ if x is honestly chosen at random. In particular, for each mode the adversary has a probability of at most $\frac{3}{4}$ to win the round, regardless of his choice of k_i and ℓ_i. We first rule out cheating on the commitments (which would allow the adversary to claim a false response as the correct one), thus losing a term $4mq_V \mathbf{Adv}_{\mathcal{A}_1}^{\mathsf{Cmt.Bind}}$ (accounting also for the commitments to x).

More formally, this is the same as assuming that the adversary now plays the game against a protocol wherein the prover has to also give the values of k_i, ℓ_i, e_i, as well as the commitments a_i, b_i, d_i *and* the randomness used for each commitment, in the first message. The prover also has to give the bits of the secret x and the randomness used to generate the commitments Com_i. The verifier does not check consistency of the responses with the responses/randomness included in the initialization. However, clearly if the prover is able to commit to one value in, for instance a_i, and then use the conjugate value with a different randomness at the end, he breaks the binding property of the commitment. We use this for our forgery in the reduction. We have to hope that the forged commitment is actually queried by the verifier and that this verifier-adversary session is successful, which is why we lose a factor $4mq_V$. This factor includes the m bit-commitments to the bits of x.

We now exclude the probability of the prover using different values that those he has committed to. Since the commitments are sent before receiving the string of modes $\{\mathsf{M}_i\}_{i=1}^m$, we argue that whatever strategy the prover employs in maliciously choosing convenient k, ℓ at each round, these values are chosen to facilitate an answer μ_i for each round i *before* knowing M_i. Depending on the choice of k and ℓ, as well as on the bits of x, the probability that μ_i is the correct answer at each round is different. We denote $\mathsf{Prob}[\mu_i = e_i] =: p_1$, $\mathsf{Prob}[\mu_i = k_i \oplus \ell_i] =: p_2$, $\mathsf{Prob}[\mu_i = e_i \oplus \ell_i] =: p_3$, and $\mathsf{Prob}[\mu_i = k_i] =: p_4$. We stress that the value of μ_i does not reflect the fact that the adversary must commit to a response before knowing M_i, but rather that the adversary must commit to a choice of k_i and ℓ_i before knowing the modes. This skews the probability that, for a certain mode, the adversary is able to give a better response and indeed maximizes the distance-fraud success probabilities. As a consequence, we are indeed able to even indicate the most successful adversarial strategy in distance fraud.

We now bound the probability that, with these values already set, the adversary can win one of the interactive rounds.

$$\begin{aligned}
\mathsf{Prob}[\mathcal{A}\text{ wins 1 round}] &\leq \sum_{k=0}^{3}\mathsf{Prob}[\mathsf{M}_i = k]\mathsf{Prob}[\mu_i = \mathsf{f}(\mathsf{M}_i, c_i)] \\
&= \frac{1}{4}\sum_{k=0}^{3}\mathsf{Prob}[\mu_i = \mathsf{f}(\mathsf{M}_i, c_i)] \\
&= \frac{1}{4}\sum_{k=0}^{3}\sum_{b=0}^{1}\mathsf{Prob}[c_i = b]\mathsf{Prob}[\mu_i = \mathsf{f}(\mathsf{M}_i, b)] \\
&= \frac{1}{8}\sum_{k=0}^{3}\sum_{b=0}^{1}\mathsf{Prob}[\mu_i = \mathsf{f}(\mathsf{M}_i, b)] \\
&\overset{1}{=} \frac{1}{8}[p_1 + p_2 + p_3 + p_4 + p_1 + (1 - p_2) + p_3 + (1 - p_4)] \\
&= \frac{1}{8}[2 + 2p_1 + 2p_3] \leq \frac{1}{8}[2 + 2 + 2] = \frac{3}{4}.
\end{aligned}$$

We remark that the equality $\overset{1}{=}$ holds due to our notation. Indeed, we have that $\mathsf{Prob}[\mu_i = \mathsf{f}(0,0)] = \mathsf{Prob}[\mu_i = \mathsf{f}(2,0)] = p_1$, $\mathsf{Prob}[\mu_i = \mathsf{f}(0,1)] = p_2$, $\mathsf{Prob}[\mu_i = \mathsf{f}(1,0)] = \mathsf{Prob}[\mu_i = \mathsf{f}(3,0)] = p_3$, $\mathsf{Prob}[\mu_i = \mathsf{f}(1,1)] = p_4$. Since $\mathsf{f}(0,1) = \mathsf{f}(2,1)$ and $\mathsf{f}(1,1) = \mathsf{f}(3,1)$, it holds that $\mathsf{Prob}[\mu_i = \mathsf{f}(2,1)] = 1 - p_2$ and $\mathsf{Prob}[\mu_i = \mathsf{f}(3,1)] = 1 - p_4$. The final inequality holds due to the fact that noting that $p_1 \leq 1$ and $p_3 \leq 1$. Indeed, the prover can maximize these probabilities by choosing $\ell_i = 0$ for all i, irrespective of the values of k_i. □

6.5.2 Resistance against Mafia Fraud

Theorem 2 (Mafia fraud resistance). *For any $(t, q_{\text{OBS}}, q_P, q_V)$-mafia-fraud adversary $\mathcal{A}$ against VSSDB, there exists: an adversary $\mathcal{A}_1$ against the hiding property of the commitment scheme* Cmt=(Commit, COpen)*, an adversary $\mathcal{A}_2$ against the unforgeability of the signature scheme $\mathcal{S} = (\mathsf{SSKGen}, \mathsf{Sign}, \mathsf{Vf})$ and an adversary $\mathcal{A}_3$ against the pseudorandomness of function* f *(in the random oracle model), such that*

$$\begin{aligned}
\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{Mafia}} &\leq \left(\frac{1}{2}\right)^m + \boldsymbol{Adv}_{\mathcal{A}_2}^{\mathsf{Unf}} + m(q_P + q_V)[4\boldsymbol{Adv}_{\mathcal{A}_1}^{\mathsf{Cmt.Hide}} + \boldsymbol{Adv}_{\mathcal{A}_3}^{\mathsf{H-PRF}}] \\
&\quad + \binom{q_{\text{OBS}} + q_V}{2}2^{-2m} + \binom{q_P + q_{\text{OBS}}}{2}2^{-3m-3|u_1|}
\end{aligned}$$

Proof. For mafia fraud resistance, we first replace the randomness w_i by truly random values, losing a term $m(q_P + q_V)\mathbf{Adv}_{\mathcal{A}_3}^{\mathsf{H-PRF}}$. At this point we can be sure that the hiding properties of the commitment does not suffer from the fact that the w_i is not chosen truly at random. The next step is to consistently replace the committed values chosen by the prover each prover-verifier and adversary-prover session by values independent of the real choice of the prover, losing the advantage of an adversary against the hiding property of the commitment, $4m(q_P + q_V)\mathbf{Adv}_{\mathcal{A}_1}^{\mathsf{Cmt.Hide}}$.

We next rule out the probability that in two different sessions, the honest verifier chooses exactly the same string of modes, losing a term $\binom{q_{\text{OBS}}+q_V}{2}2^{-2m}$. Furthermore, we exclude the possibility that the prover chooses the exact same randomness and the same shares k and ℓ in two sessions. We lose a term $\binom{q_P+q_{\text{OBS}}}{2}2^{-3m-3|u_1|}$. As a consequence, for each verifier-adversary session there is at most one adversary-prover session sharing the same mode values (this is a session in which the adversary forwards the same modes seen before). The next step is to rule out that the adversary is able, for the two concurrent sessions sid and sid^*, the first one being a verifier-adversary session while the second one is an adversary-prover session, to change any of the verifier's input and yet be able to forge the signature at the end. We lose here a term $\mathbf{Adv}_{\mathcal{A}_2}^{\mathsf{Unf}}$.

Now the adversary's only solution is to guess either: (a) the values of all the challenges (which happens with probability 2^{-m}), (b) the value of the bits of the key x (which happens with probability 2^{-m}, but in this case the adversary also needs to forge the signature or the values of all the pertinent witnesses for the verification), (c) the values of all the responses (with probability 2^{-m}). Combining all these results this leads to the indicated bound. □

6.5.3 Resistance against slow phase impersonation

Theorem 3 (Slow phase impersonation resistance). *For any $(t, q_{\text{OBS}}, q_P, q_V)$-impersonation adversary $\mathcal{A}$ against VSSDB, there exists an adversary $\mathcal{A}_1$ against the unforgeability of the signature scheme $\mathcal{S} = (\mathsf{SSKGen}, \mathsf{Sign}, \mathsf{Vf})$ such that:*

$$\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{ImpSec}} \leq \binom{q_{\text{OBS}} + q_V}{2}2^{-2m} + \binom{q_P + q_{\text{OBS}}}{2}2^{-3m-3|u_1|} + \boldsymbol{Adv}_{\mathcal{A}_1}^{\mathsf{Unf}}.$$

Proof. We first rule out the possibility that the verifier chooses the same modes in two different sessions, and then the possibility that the prover chooses the same shares and randomness for two different sessions, thus losing the same two terms as for mafia-fraud resistance. Thus, any verifier-adversary session the adversary opens has at most one adversary-prover session for which it shares modes and commitments. At this point, the definition requires that the adversary wins without producing the same transcripts with the concurrent adversary-prover session. We upper-bound this probability by observing that any change in the transcript requires the adversary to be able to forge the signature, thus obtaining the claimed bound. □

6.5.4 Resistance against Terrorist fraud (KeyTF-security)

Theorem 4 (KeyTF-Security). *The VSSDB protocol is KeyTF-secure.*

Proof. We prove this statement in two steps. We first consider a successful terrorist fraud adversary $\mathcal{A}$. Our goal is to show that either this adversary has a negligible success probability or that he is helpful to a mafia-fraud adversary $\mathcal{A}'$. In the first step of the proof, we show how the adversary $\mathcal{A}'$ reconstructs a guess of the secret x by running $\mathcal{A}$ as a black box. In the second stage of the proof, we show that this gives $\mathcal{A}'$ an advantage over any other mafia-fraud adversary.

We consider an adversary $\mathcal{A}'$ who runs $\mathcal{A}$ internally, and who ends up with a guess $\hat{x}$ of x. For each verifier-adversary session of the terrorist-fraud adversary $\mathcal{A}$, the adversary $\mathcal{A}'$ queries $\mathcal{A}$ for each fast round and each of the two challenges ($\mathcal{A}'$ does not change the mode). If $\mathcal{A}$ answers on both branches, $\mathcal{A}'$ sets $\hat{x}_i$ to be the XOR of the two responses. If the adversary does not respond on at least one branch, $\mathcal{A}'$ sets $\hat{x}_i$ to be a randomly-chosen bit. The following statements hold.

- $\mathcal{A}$ knew both bits correctly. In this case, $\mathcal{A}$ always passes the round, but the guessed bit $\hat{x}_i$ is also correct.

- $\mathcal{A}$ gets exactly one bit correctly. In this case, $\mathcal{A}'$ has the wrong bit. On the other hand, the adversary fails the session with probability $\frac{1}{2}$.

- $\mathcal{A}$ gets both bits wrong, in which case $\mathcal{A}$ fails the session, but $\mathcal{A}'$ gets the right bit.

- $\mathcal{A}$ refuses to answer at least for one branch. At this point the adversary has a probability of at least $\frac{1}{2}$ to fail, while $\mathcal{A}'$ guesses the corresponding guessed bit of x is $\frac{1}{2}$.

In fact the number of bits of x that the adversary $\mathcal{A}'$ expects to retrieve is $\mathcal{A}$'s success probability to succeed in the fast rounds. The adversary $\mathcal{A}'$ does the same thing for each of the q_V sessions $\mathcal{A}$ has with the verifier. Thus, $\mathbf{Leak}_{\mathcal{A}'}(x||\mathsf{sk}_{\mathsf{Sign}}) = \mathbb{P}[\mathcal{A}\ wins]$.

In contrast, for any mafia-fraud resistant adversary, the expected number of retrieved bits is 0, since with high probability nor the signatures, neither the committed values cannot be forged. In addition, since the modes and commitments are picked at random by the honest prover and the honest verifier, it is highly unlikely to have two sessions with exactly the same committed values, and for which at least one round of the distance-bounding phase have the same modes with different challenges.

We now show that $\mathcal{A}'$ is more successful than any mafia-fraud adversary $\mathcal{B}$. If $\mathcal{B}$'s strategy is to change any of the input between the prover and the verifier during that session, then $\mathcal{A}'$ runs the first part of the protocol (which depends on x) using his guess of it in every session with the verifier, succeeding with the same probability as $\mathcal{A}$, and uses the same strategy to forge the signature as $\mathcal{B}$ does. □

6.5.5 Introducing cheat modes to tackle terrorist fraud security (GameTF security)

In order to ensure terrorist-fraud resistance, we have to build a backdoor into the protocol. To do this, we use an idea similar to that proposed by Fischlin and Onete in [FO13b]. However, we cannot directly import their solution as our protocol is asymmetric, whereas the original trick assumes that the verifier knows the prover's secret key. Instead, we take an approach similar to all-or-nothing security and modify the protocol in two steps. More precisely during the first step, if the prover receives a message starting with a 1 instead of the first protocol message $(0|N_V)$, he will compare the remainder of the message with the secret key x. If the received message is sufficiently close, he returns the full key x as well as the witnesses $(H)^i(x)$ that were used by the TTP to compute the commitments Com_i as depicted in Protocol 4.

Any party, including an adversary, can run this strategy. However, note that the protocol does not return the secret signature key of the prover, thus on its own knowing x will not significantly help an adversary to authenticate due to the final signature. To enable a successful terrorist-fraud adversary to authenticate afterwards without the prover's help, we also change the authentication protocol. More precisely, we introduce a "cheat" running mode, in which a party can use the values of x and v_i to authenticate just by echoing the received challenges. As an honest prover only discloses the values of x and v_i in exchange for a good guess of the secret key x, these values will not occur in observed honest-prover to honest-verifier communication. We also prevent distance fraud, as using this strategy will reduce the success probability of a distance-fraud adversary to just $\frac{1}{2}$ per round (the probability of guessing the challenges in advance). However, if a malicious prover helps a MIM adversary to authenticate during terrorist fraud, the adversary will learn a significant part of the secret key x. This will enable the terrorist-fraud MIM adversary to learn (with high probability) the values of x and v_i, which in turn enable him to make the verifier run the protocol in cheat mode, as depicted in Protocol 5.

Theorem 5. *GameTF-Security The VSSDB protocol is* GameTF*-secure.*

Proof. The first part of the proof goes as for the VSSDB protocol. We conclude that the mafia fraud adversary $\mathcal{A}'$ gains as many bits of x as is the terrorist adversary $\mathcal{A}$'s probability to win. We denote by $p_{\mathcal{A}}$ the probability that $\mathcal{A}$ wins. Then the cheating mode is activated with exactly this probability. In this case, one the two following events holds:

- $p_{\mathcal{A}}$ is negligible, in which case we dismiss him.
- $p_{\mathcal{A}}$ is non-negligible and thus larger than the (negligible) mafia-fraud resistance of the protocol.

□

6.6 Improving the response functions

In section 6.4, we have designed a distance-bounding protocol that uses modes in conjunction with the default challenge bit scheme that is usual to distance-bounding protocols. We introduce the notion of modes to prevent a malicious prover from selecting the values k_i and ℓ_i in order to maximize his success probability in a distance fraud like described in section 6.2.2. In fact, during the initialization phase, the prover commit to the x_i, k_i and ℓ_i. Then, the verifier announces the mode for each round of the distance-bounding phase. Provided that the modes are chosen at random by the verifier, the adversary cannot reply in advance during the distance-bounding phase because the responses bit of the modes are never the same regardless of the value of x_i, k_i and ℓ_i chosen in advance. Therefore, for each round of the protocol, the distance adversary may receive the challenges from the verifier before sending the corresponding response to the verifier. To succeed a distance faud, the adversary has two options: (i) he guesses each mode of the distance-bounding phase in advance with probability $\frac{1}{2}$ of success. (ii) if the bit $x_i = 0$ then he chooses $k_i = \ell_i = x_i = 0$ then regardless of the modes, he win the round with probability 1 (see the first line of the table 6.6.

From the point (ii) we see that the advantage of an adversary to succeed a distance fraud can be correlated to the number of bits of his secret key equal to 0. In the latter we refer to this type of key (with as many bits equal 0 as *weak key*).

Protocol 3: Verifiable Secret Sharing and Distance-Bounding (VSSDB).

Verifier V		**Prover** P
Public key $\mathsf{pk} = \{\mathsf{Com}, \mathsf{pk}_{\mathsf{Sign}}\}$ with $\mathsf{Com} := (\mathsf{Com}_1, \ldots, \mathsf{Com}_m)$ Private encryption key: sk_{V}		Private key $x, \mathsf{sk}_{\mathsf{Sign}}, \{\nu_1, \ldots \nu_m\}$
	Initialization phase	
		For $i = 1$ to m generate: $k_i, \ell_i \overset{\$}{\leftarrow} \{0,1\}$ $e_i = \mathcal{E}_{k_i \Vert \ell_i}(x_i)$ $u_i, v_i \overset{\$}{\leftarrow} \mathbb{Z}_n^*$ $w_i = u_i^{-1} v_i^{-1} \mathsf{H}^i(x) \mod n$ $a_i = \mathsf{Commit}(k_i, u_i)$ $b_i = \mathsf{Commit}(\ell_i, v_i)$ $d_i = \mathsf{Commit}(e_i, w_i)$
	$\xrightarrow{0 \Vert N_{\mathsf{V}}}$	
		Set: $m_{\mathsf{P}} := \{a_i, b_i, d_i\}_{i=1}^m \Vert N_{\mathsf{V}}$ Compute: $\mathsf{c}_{\mathsf{P}} = \mathsf{Enc}_{\mathsf{pk}_{\mathsf{V}}}(m_{\mathsf{P}} \Vert \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, m_{\mathsf{P}}))$ Set: $\mathcal{W} := \{x, \{k_i, \ell_i, e_i, u_i, v_i, w_i,$ $a_i, b_i, d_i, \nu_i\}_{i=1}^m, \mathsf{sk}_{\mathsf{Sign}}\}$ $\pi := \mathsf{Prove}\{\mathcal{W} : \mathsf{c}_{\mathsf{P}}$ well formed$\wedge$ consistent with $\{\mathsf{Com}_i\}_{i=1}^m\}$
	$\xleftarrow{\mathsf{c}_{\mathsf{P}}, \pi}$	
Decrypt c_{P} to $m' \Vert N_{\mathsf{V}}^* \Vert \sigma^*$; If π, σ^* do not check or $N_{\mathsf{V}}^* \neq N_{\mathsf{V}}$, abort. **For** $i = 1$ to m generate $\mathsf{M}_i \overset{\$}{\leftarrow} \{0,1\}$	$\xrightarrow{\{\mathsf{M}_i\}_{i=1}^m}$	
		Parse as: $\hat{\mathsf{M}}_1, \ldots \hat{\mathsf{M}}_m \in \{0,1\}$ For $i = 1$ to m, compute $\mathsf{f}(\hat{\mathsf{M}}_i, 0), \mathsf{f}(\hat{\mathsf{M}}_i, 1)$, $\gamma(\hat{\mathsf{M}}_i, 0)$, and $\gamma(\hat{\mathsf{M}}_i, 0)$ as below: $\mathsf{f}(0,0) = \leftarrow e_i$ $\mathsf{f}(0,1) = \leftarrow k_i \oplus \ell_i$ $\mathsf{f}(1,0) = \leftarrow k_i$ $\mathsf{f}(1,1) = \leftarrow e_i \oplus \ell_i$ $\gamma(0,0) = \leftarrow w_i$ $\gamma(0,1) \leftarrow (k_i, u_i, \ell_i, v_i)$ $\gamma(1,0) \leftarrow u_i$ $\gamma(1,1) = \leftarrow (e_i, w_i, \ell_i, v_i)$
	Interactive phase **For** $i = 1$ to m	
Picks $c_i \overset{\$}{\leftarrow} \{0,1\}$	$\xrightarrow{c_i}$	Parse as $\hat{c}_i$
Store r_i and RTT δt_i	$\xleftarrow{r_i}$	$r_i \leftarrow \mathsf{f}(\hat{\mathsf{M}}_i, \hat{c}_i)$
	Verification Phase	
	$\xleftarrow{\{\gamma(\hat{\mathsf{M}}_i, c_i)\}_{i=1}^m}$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, \{\mathsf{M}_i\}_{i=1}^m \Vert \{c_i, r_i\}_{i=1}^m \Vert N_{\mathsf{V}})$
	$\xleftarrow{\sigma}$	

Verify σ, open commit for $\mathsf{f}(\hat{\mathsf{M}}_i, \hat{c}_i)$ with $\gamma(\hat{\mathsf{M}}_i, \hat{c}_i)$, check $\delta t_i \leq t_{\mathsf{max}}\ \forall i$. Accept iff. all checks succeed.

Protocol 4: All-or-nothing key disclosure.

Party Prt		**Prover** P
Input $\hat{x}$		Private key $x, \mathsf{sk}_{\mathsf{Sign}}, \{\nu_1, \ldots \nu_m\}$
	$\xrightarrow{1\|\|\hat{x}}$	
		With probability $2^{-(x \oplus \hat{x})}$ do: Set: $\mathsf{Keys} := \{x, \mathsf{sk}_{\mathsf{Sign}}, \{\nu_1, \ldots, \nu_m\}\}$.
	$\xleftarrow{\mathsf{Keys}}$	

Protocol 5: Terrorist VSSDB.

Verifier V		**Prover** P
Public key $\{\mathsf{Com}, \mathsf{pk}_{\mathsf{Sign}}\}$ with $\mathsf{Com} := (\mathsf{Com}_1, \ldots, \mathsf{Com}_m)$ Private encryption key: sk_{V}		Private key $x, \mathsf{sk}_{\mathsf{Sign}}$
	Initialization phase	Initialize as in Figure 3.
	$\xrightarrow{0\|\|N_{\mathsf{V}}}$	
		If first bit is 0, run protocol in Protocol 3. Else react as in Figure 4 and halt.
	$\xleftarrow{0\|\mathsf{c_P}, \pi}$	
If first bit is 0, run protocol as in Figure 3. Else: parse message as $\hat{x}, \{\hat{\nu_i}\}_{i=1}^m$ If $\exists i$ such that $\mathsf{Commit}(\hat{x_i}, \hat{\nu_i}) \neq \mathsf{Com}_i$ reject P. Else, set $\mathsf{SBit} = \texttt{cheat}$.		
	Interactive phase **For** $i = 1 \cdots m$	
Pick $c_i \overset{\$}{\leftarrow} \{0, 1\}$	$\xrightarrow{c_i}$	
Store r_i and RTT δt_i	$\xleftarrow{r_i}$	
	Verification Phase	
Accept if and only if for all i, $r_i = c_i$ and $\delta t_i \leq t_{\mathsf{max}}$.		

x_i	ℓ_i	e_i	$k_i \oplus \ell_i$	k_i	$e_i \oplus \ell_i$
0	0	0	0	0	0
0	0	1	1	1	1
0	1	1	1	0	1
0	1	0	0	1	1
1	0	1	0	0	1
1	1	0	1	0	1
1	0	0	1	1	0
1	1	1	0	1	0

Table 6.1: Truth table representing the response function with two modes.

One solution to rule out such possibility is to assume that the keys are always generated by the TTP honestly, thus ensuring that each bit of the key is 0 with probability $\frac{1}{2}$ (independently). This assumption is used in the bound for our distance-fraud resistance above. While this assumption is sufficient in theory, in some case, we would like that the prover register a key without revealing the actual key to the TTP. Therefore a second response function is needed.

In the following, we construct an improved response function that resolves the problem mentioned previously (it does not rely on the assumption that the TTP generates the secret key for the user). Therefore, any party can register a key directly with the CA.

The new response function we propose, is based on the default response function. For the mode 0 and 1 we keep the same behaviour as the default response function. We create the two remaining modes by flipping (the bit is x-ored with the value 1) the bit of one response-bit out of the two available.

$$\mathsf{f}(\mathsf{M}_i = 0, c_i) = \begin{cases} e_i & \text{if } c_i = 0 \\ k_i \oplus \ell_i & \text{if } c_i = 1 \end{cases} \qquad \mathsf{f}(\mathsf{M}_i = 2, c_i) = \begin{cases} e_i & \text{if } c_i = 0 \\ \overline{k_i \oplus \ell_i} & \text{if } c_i = 1 \end{cases}$$

$$\mathsf{f}(\mathsf{M}_i = 1, c_i) = \begin{cases} e_i \oplus \ell_i & \text{if } c_i = 0 \\ k_i & \text{if } c_i = 1 \end{cases} \qquad \mathsf{f}(\mathsf{M}_i = 3, c_i) = \begin{cases} e_i \oplus \ell_i & \text{if } c_i = 0 \\ \overline{k_i} & \text{if } c_i = 1 \end{cases}$$

Then during the verification phase, the prover reveals the value $\gamma(M_i, c_i)$ in which M_i reprensents the modes and c_i the challenge received from the verifier.

$$\gamma(\mathsf{M}_i = 0 \text{ or } 2, c_i) = \begin{cases} w_i & \text{if } c_i = 0 \\ k_i, u_i, \ell_i, v_i & \text{if } c_i = 1 \end{cases} \qquad \gamma(\mathsf{M}_i = 1 \text{ or } 3, c_i) = \begin{cases} e_i, w_i, \ell_i, v_i & \text{if } c_i = 0 \\ u_i & \text{if } c_i = 1 \end{cases}$$

Therefore, we have always the condition that revealing the response bits of one mode reveals the corresponding bit of the secret key. By forcing one response-bit to its opposite value (x-ored with 1), we rule out the possibility that the choice of the value $x_i = k_i = \ell_i$ has incidence on the construction of the response-bits .

x_i	ℓ_i	e_i	$k_i \oplus \ell_i$	$\overline{k_i \oplus \ell_i}$	$e_i \oplus \ell_i$	k_i	$\overline{k_i}$
0	0	0	0	1	0	0	1
0	1	1	1	0	0	0	1
0	0	1	1	0	1	1	0
0	1	0	0	1	1	1	0
1	0	1	0	1	1	0	1
1	1	0	1	0	1	0	1
1	0	0	1	0	0	1	0
1	1	1	0	1	0	1	0

Table 6.2: Truth table representing the response function with four modes.

From the table 6.6, we see that with any value for x_i, k_i, ℓ_i the adversary can maximize his success probability for at most two modes out of the four availables. As the modes are selected at random by the verifier, and the prover does not know them during the commitment phase, then the adversary success probability is equivalent to the probability to guess the mode in advance that is $\frac{1}{2}$.

6.7 Conclusion

In this chapter, we have presented a DB protocol called VSSDB, which is asymmetric (*i.e.*, the prover and verifier do not share secrets) while provably resisting terrorist-fraud attacks (as captured by the GameTF flavor of terrorist-fraud resistance [FO13b]). One of the main novelty of our protocol is the use of *modes*, which are communicated during slow phases, and complement the challenges sent during the time-critical rounds. More precisely, the responses answered by the prover during one of these rounds depend both on the mode of this round *and* the challenge. This feature improves the protocol's security, while preserving the lightweightness of calculations performed during the time-critical rounds. In addition, in contrast to other DB protocols, we rely on the use of *three* shares rather than two, which allows for more richness in the responses while better hiding the secret. In particular, by using a homomorphic commitment scheme and *bitwise* verification of the commitment, we essentially prevent the attack of Bay and co-authors [BBM$^+$12] against DBPK-Log scheme on which VSSDB builds. We also rely on the countermeasure of signing the entire protocol transcript at the end to ensure strong mafia-fraud resistance. Finally to achieve GameTF-security, we add a cheating mode that relies on a trick proposed by Fischlin and Onete [FO13b]. In a nutshell, if the adversary can supply a close-enough guess of the secret key x, the prover will return the correct secret key x as well as the witnesses that were used to generate the commitments of each bit of x that the verifier has. By using these values, the adversary can then authenticate to the verifier.

PROPS: Un système de preuves de localisation respectueux de la vie privée

Comme énoncé précédemment, les services basés sur la localisation doivent répondre à de nouvelles exigences sécuritaires, à savoir garantir l'authenticité de la position annoncée par un utilisateur sans la disponibilité au préalable d'une architecture de confiance. En se basant sur le modèle de sécurité présenté au chapitre 4, nous proposons un système de preuve de localisation baptisé PROPS (*Pivacy-pReserving lOcation Proof System*).

PROPS emploie une approche collaborative au problème des preuves de localisation, permettant ainsi à des dispositifs voisins de se générer mutuellement des preuves de localisation. Plus précisément, le protocole se déroule en trois phases: la phase d'enregistrement, la phase de collecte et la phase de vérification. Lors de la phase d'enregistrement, chaque utilisateur reçoit une clé secrète certifiée par une autorité de certification. Cette clé secrète est unique à chaque utilisateur et nous supposons qu'elle n'est jamais partagée avec d'autres entités. Ensuite, lors de la phase de collecte, le prouveur demande aux témoins présents dans son voisinnage immédiat de lui signer sa position (au format encodé) et une mise en gage sur son identité secrète. Chaque témoin désireux de participer à la création de la preuve, procède à la vérification de la présence physique du prouveur dans son voisinnage immédiat, auquel cas, il lui génère sa preuve de localisation sinon la phase de collecte est annulée et possiblement reportée à une autorité de gestion de fraudes. La vérification de la position du prouveur est réalisée à l'aide d'une primitive appelée ProximityTesting construite à partir d'un protocole délimiteur de distance (voir Section 7.3 pour plus de détails). Finalement, lors de la phase de vérification des preuves, le prouveur présente ses preuves de localisation au vérifieur accompagnée d'une preuve de connaissance de l'identité secrète contenu dans les mise en gage et une granularité liée à la position encodée. Les propriétés de sécurité que nous garantissons par le biais de notre protocole sont les suivantes: la non-transférabilité des preuves, la non-forgeabilité de celles-ci et la résistance contre les attaques de localisation lors de la phase de collecte (voir Section 7.5 pour plus de détails). Du point de vue de la protection de la vie privée, nous garantissons l'anonymat de l'identité secrète du prouveur et des témoins durant la phase de collecte et la phase de vérification. Nous garantissons aussi la non-chaînabilité de leurs actions à travers le système (voir Section 7.5 pour plus de détails).

Chapter 7

PROPS: a PRivacy-preserving lOcation Proof System

7.1 Introduction

A secure LBS requires that a mobile user certifies his position before gaining access to a resource. This is especially the case for location sensitive application like location-based access control and defining location-aware security policies (*e.g.*, this laptop should not be taken out of this building, this file should not be opened outside of a secure room, etc..). Currently, most of the existing solutions addressing this issue do not offer enough security or privacy guarantees to their user (see Chapter 3). In this chapter, we describe our protocol for a secure and decentralized location proof system. Our design aims to satisfy the security and privacy properties we have defined in Chapter 3. We also rely on the concept of distance bounding, to provide a method of detecting various situation of localization attacks. In fact, we have shown in Chatper 5 that a distance-bounding protocol is an accurate method of gauging a prover's presence within range of the witness.

The structure of this chapter is as follows: in Section 7.2, we describe how users interact within the system and describe the system assumptions. Then, in section 7.3, we present the cryptographic primitives we used in the design of our protocol. In Section 7.4, we describe the process for users of collecting location shares from their witnesses and how verifier can check for the validity of location proofs. Next, in Section 7.5 we analyse the security and privacy properties of our proposal. Section 7.6 reviews the implementation and performance of our protocol. Finally, in Section 7.7 we summarize this chapter.

7.2 Preliminaries

In this section, we discuss the technical information pertaining to PROPS. More precisely, we give a global overview of the architecture, then we define the algorithms used by the system before presenting the system assumptions. We discuss the relevant threats that PROPS must protect against and the security properties required in order to meet our definition of a complete and secure protocol. Finally, we briefly outline the role of the verifier within the protocol.

7.2.1 User interactions

Our system consists of a set of mobile users equipped with location enabled devices that are all synchronized according to a global clock. Each mobile device is capable of obtaining his current absolute position thanks to a positioning system and we suppose that these systems are interoperable. For sake of simplicity, let be a user P visiting a location L at time T. P can request a location proof from other users present at the same location L during time T. During this interaction, the device of P plays the role of the *prover* while the other devices that collaborate to create the proof are called the *witnesses*. The first step of the protocol is for the prover to broadcast a request (location proof request) stating his current secure pseudonym, the claimed location and the current time to potential witnesses present at the same location. When a surrounding user accepts to testify the presence of the prover, the latter becomes a *witness* for that session. Then, the rest of the procedure consists in a secure bipartite computation between the prover and that witness during which the witness is responsible for verifying the correctness of the spatio-temporal information claimed by the prover; that is he is indeed at the location L close to the witness at global time T. To counter localization attacks (see Chapter 2), the prover engages in a distance-bounding protocol (see Chapter 5 and Chapter 7) with each of the witnesses to ascertain that he is within proximity of that device, thus proving himself to that particular proof provider. It is assumed that the distance-bounding process employed will provide an accurate reading of the proximity of the prover to a witness. The ability of distance-bounding to accurately place a device in an area is an underlying security property of PROPS and is required for its correct functionality. It is also assumed that proxy/terrorist fraud attacks will be detected through the use of the distance-bounding protocol employed. When the prover behaves honestly, he obtains a *location proof share* (LPS) from his witness. The location proof share contains the cryptographic pseudonym of the prover, the results of the distance bounding exchanges and the encryption of the spatio-temporal information (location and time). At this stage, the location proof shares can be aggregated to create a *location proof* (LP). A location proof may be used by the prover to convince a *verifier* (for example police authorities, location-based service, etc.) of his presence at the location L. The verifier then makes his decision based on this proof information. This situation is illustrated in Figure 7.1. Users are registered to the system by a trusted central authority called the *certification authority* (CA). CA is the trusted third party responsible for issuing the credentials to newly registered users. These credentials can be considered as being the “secret identity” of these users. The cryptographic pseudonyms are derived from the secret key of the user. The *anonymity lifter* (AL) is the trusted party that has the capacity to lift the anonymity of a particular user from a location proof when needed (for instance upon request from a judge).

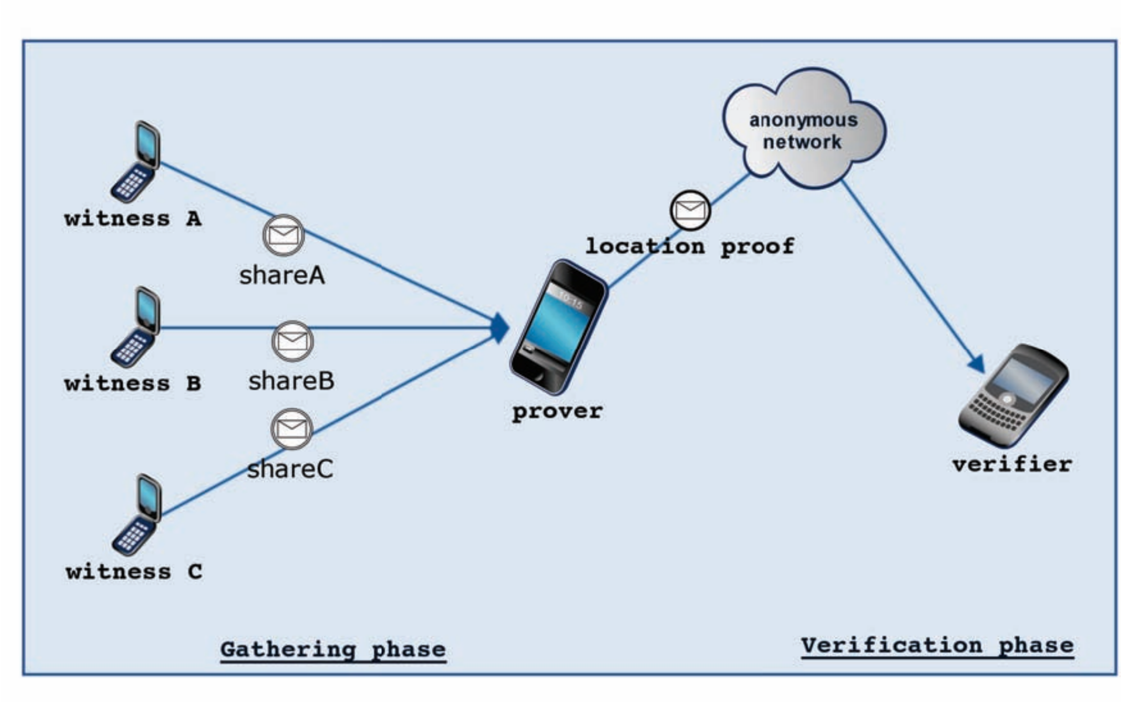

Figure 7.1: Global overview and data flow of Props.

7.2.2 Definition of the algorithms

The architecture of Props is summarized in Figure 7.1 and consists of the following four different procedures.

- *Initialisation.* During the set up of the location proof system, the Certification Authority (CA) runs the Setup algorithm to generate the group public parameter $gpk = (VKG, VKCert)$ in which VKG represents the public verification key for the unique group signature and $VKCert$ the public verification key for CL-signature certificate issued by the CA. It should also be possible for the CA to generate issuance keys that will be used by authorised issuer party to dynamically add users to the system. The CA also decides at that time on the values of parameters such as m, the maximum number of malicious users tolerated by the system and k the minimum number of different location shares needed to generate a valid location proof certificate. By assumption, we always have $m < k$.

- *Join.* When a user want to join the system, he needs to run the procedure VRegister together with the CA or an authorized Issuer to obtain his anonymous credentials $usk = (SKG_u, S_u, cert_{S_u})$ in which SKG_U represents his private key for unique group signature, S_u the secret identifier of the user and $cert_{S_u})$ a certificate over S_u made by the certification authority or an authorized issuer such that the user can later prove that he has a valid CA-certificate on his secret identifier in a zero-knowledge way.

- *Location proof gathering.* When a prover wants to obtain a proof of his current location, he runs the procedure LTestify in collaboration with neighbouring users (*i.e.*, witnesses) to obtain location shares. After collecting at least k shares from different witnesses, the prover can produce a location proof by running the algorithm LProve.

- *Location proof verification.* During this step, the prover run Vf with a verifier to prove to the latter that he was at a specific location at a particular moment in time. In our case, this verification process (*cf.* Section 7.4.2) requires a proof of ownership by the current prover and also a check from the verifier that the location proof of the previous step contains at least k shares from different witnesses. It is possible to envision that if the verifier suspects a cheating attempt from the current prover (*e.g.*, if the location proofs is composed of two or more shares generated by the same witness), then he may forwards the fraud evidence to the Anonymity Lifter (AL) who has the capacity to reveal the identity of the cheating witnesses.

7.2.3 System Assumptions

Within the architecture of Props, we make the following system assumptions.

Positioning capability. Users are mobile entities capable of positioning themselves into space. For instance, a user might be able to localize himself by using his GPS or through the help of a dedicated positioning infrastructure. We assume the imprecision of the location computed to be negligible.

Synchronized clocks. Each user possesses a local clock on his device that is synchronized with the clocks of other users. Thus, when two users communicate together, they rely on the same time referential. In practice, users can use the clocks of their GPS or GSM device.

Anonymous communication channel. Provers can broadcast message to neighboring witnesses without disclosing identifying information (*e.g.*, MAC or IP addresses).

7.3 Building blocks

We assume that users of Props have access to standard cryptographic tools such as (pseudo-)random number generators, hash functions and public key signature schemes.

However, our protocol also relies on other cryptographic primitives that we use as building blocks. These primitives are detailed thereafter.

7.3.1 Unique group signatures

Group signature schemes have been introduced by Chaum and van Heyst to provide anonymity to the signer of a message [CvH91].

In fact, there is a single public verification key for the group, but each member of the group receives a different private signing key from the group manager (who could be for instance the CA). A group signature scheme (with optional anonymity removing) consists in general of the four following operations.

- *Registration of the user.* During the Join operation, the CA assigns to the user a new private signature key, which we denote by SKG_U.
- *Signature of a message in behalf of the group.* The SignGroup operation takes as input a message m and a signing key SKG_U and produces a signature $\sigma_{G,U}(m)$ of the message m.
- *Verification of a group signature.* The VerifySignGroup operation allows to check the validity of a group signature. It requires as input a verification key for the group VKG, which has been setup by the CA and is publicly known, as well as a message m and a group signature on this message $\sigma_{G,U}(m)$. VerifySignGroup produces a binary output : accept or reject depending on the validity of the signature.
- *Anonymity removing.* From the point of view of the verifier, it is impossible to distinguish if two group signatures come from the same individual or not. However in exceptional situations, the AL can (in association with the verifier) retrieve the identity of a particular signer via the LiftAnonymity operation. This operation takes as input a message m and a group signature on this message $\sigma_{G,U}(m)$ and produces as output the identity of the signer U. In practice, this is often done by first finding the corresponding signature key SKG_U and then retrieving the identity associated to this key.

Group signatures provide multiple-show unlinkability in the sense that, from the point of view of the verifier, it is impossible to distinguish if two group signatures originates from the same individual or not. However, in some specific cases, such as the situation in which the same message is signed twice, it might be interesting to be able to link the signatures performed by the same user. For instance, in the context of location proof we want to ensure that all the shares of a location proof are signed by different unique witnesses (*uniqueness property*). In order to enforce this property, we use an extension of group signatures known as *unique group signature* [FZ12]. More precisely, in a unique group signature if a signer produces two different signatures of the same message (*i.e.*, two location shares of the same proof), then both signatures while randomized will always have a large common component with high probability. This component can act as a unique identifier in order to link these signatures. Therefore, unique group signatures are usually equipped with a detection algorithm that can infer if two signatures on the same message have been issued by the same signer, and raises an alarm if this situation occurs.

7.3.2 Commitment schemes

Commitment schemes [GS07, DF02] arise from the need for parties to hide from the others a choice until they decide to reveal that choice in such a way that it is fair to all parties. The main problem here is that we do not want one party to find out about any party's commitment before the latter opens this value (*hiding* property). On the other hand, we do not want a party to be able to open its commitment in multiple ways (*binding* property).

More formally, a commitment scheme consists of a triple of algorithms (Setup, Commit, Open) that are respectively used during the initialization of the commitment scheme, the commit phase and the opening phase. For an example of how a commitment scheme works, let consider Bob who wants to send a commitment to Alice. At the beginning of the protocol, the commitment scheme may be initialized by a trusted party that we called Oscar using the Setup algorithm.

During the commit phase, Bob uses the algorithm Commit that takes as input a private input m and a random string r, then the algorithm outputs a value C called the commitment, (*i.e.*, $C \leftarrow \mathsf{Commit}(m, r)$). Actually, the value C is revealed to Alice.

Later, when Bob wants to open the commit C to Alice, he sends the value r. Then Alice runs algorithm $\tilde{m} \leftarrow \mathsf{Open}(C, r)$ and accepts if the value $\tilde{m} \neq \perp$. This step is called the opening phase.

7.3.3 Zero-knowledge proof

Zero-knowledge proof [GMR85] is a protocol enabling a party (the prover) to convince another party (the verifier) about the validity of a statement without revealing any information apart from the veracity of the statement itself. If proving the statement requires knowledge of some secret information on the part of the prover, the definition implies that the verifier will not be able to prove the statement in turn to anyone else, since the verifier does not possess the secret information.

Indeed, a zero-knowledge protocol may not allow the verifier to learn any computational information about the secret input of the prover. Let W denote a boolean statement, (*i.e.*, a function that on input α either outputs 1 (true) or 0 (false)). Suppose a prover P that knows some valid input string w, (*i.e.*. . . , w is valid iff $W(w) = 1$. P wants to convince V that he knows some valid input without revealing that value in fact to V). Such a zero-knowledge protocol can be denoted by $\mathsf{ZKProof}\{(w) : W(w) = 1\}$. At the end, if the protocol outputs true, then V can be assured that P knows a string w' such that $W(w') = 1$. In fact, our notational convention is that the elements listed in the brackets denote knowledge of which is being proved and not known to the verifier. A zero-knowledge proof must satisfy at least the three following fundamental properties, which are *completeness*, *soundness* and *zero-knowledge*.

The completeness property means that if the statement is true then a verifier who follows the recipe of the protocol will always be convinced provided that the prover knows the secret input. The soundness property ensures that if the statement is false then the prover cannot convince the verifier about the validity of the statement, except with a negligible probability. Finally, the zero-knowledge property guarantees that the interactions between the prover and the verifier do not convey any information about the secret information except its validity.

Possible application of zero-knowledge proofs is the design of privacy-preserving identification schemes and anonymous credentials [FS87, FFS88, GQ88, BCL06].

7.3.4 Proximity testing protocol

We assume that each user has access to a *proximity testing protocol*, which will be used to convince the witness that the prover is indeed located within some range. We do not make any *a priori* assumptions about the technology used to implement this protocol, but rather we will use it as a black-box during the execution of the location proof protocol. However, we are conscious that the choice of a particular implementation of a proximity testing protocol (as well as the underlying technology) will influence the resulting security of the architecture. For instance, most of the current proximity testing protocols robust against the proxy attack only exist currently as research prototypes. Distance bounding (see Chapter 6) is employed within this research to provide proof of a device's presence in the area for use by the location proof system. The technique allows the witness to distinguish whether the prover is physically present within the claimed area.

Implementing the Bussard and Bagga protocol

Here, we describe how to incorporate a distance bounding protocol in our setting. Mainly, we use the Bussard and Bagga [BB05b] because it is asymmetric and preserves the anonymity of the prover but our architecture is actually agnostic to the DB protocol used.

The proximity testing procedure consists of three phases. The first phase is the preparation one, in which the prover encrypts his private key S_U with a random symmetric key k and gets the corresponding encrypted message e. Then, the prover commits individually to each bit of e and k, which results in two sequences of bit commitments R_0 and R_1. This phase can be performed offline by the prover to save time.

During the second phase, the prover sends R_0 and R_1 to the witness, which then starts a multi-round fast-bit-exchange with the prover. In each round i, the witness sends a challenge bit $b_i \in \{0, 1\}$, to which the prover replies with the i-th bit of R_b. Since the witness never learns both bit values, he will also never learn the secret S_U. After the multi-round fast-bit-exchange, the witness verifies the corresponding bit commitments of R_0 and R_1 (only for the received bits) by asking the prover to provide the opening information for these commitments.

During the third phase, the values R_0 and R_1 are used by the witness to derive a pseudonym C_2. Finally, the prover convinces the witness that C_2 and C_1 correspond to Pedersen commitments on the same value through a zero-knowledge proof [CL03]. For more details about how the pseudonym C_2 is constructed, we refer the reader to [BB05b].

In contrast to the original protocol from [BB05b] and its implementation in STAMP [WZP$^+$], the witness and the verifier do not need to have a public key to authenticate the prover. Instead the authentication is performed using pseudonyms and zero-knowledge proofs to preserve the privacy of the prover. Revealing the values k and e to a colluder also disclose the long-term secret of the prover, thus ensuring that the proximity testing procedure is resistant to terrorist frauds.

Thereafter, we refer to the black-box primitive executing the proximity testing protocol as $\mathsf{ProximityTesting}(\Delta)$, in which Δ represents the distance threshold of the protocol. This primitive outputs accept if the prover is indeed within Δ range of the witness and reject otherwise.

7.3.5 Hash chains

A *hash chain* $\mathcal{C}$ is a set of values $x_0, \ldots, x_n$ where $x_i = h(x_{i-1})$ such that h is a cryptographic hash function, $i \in [1, n]$ and x_0 is a valid input for h. The length of a hash chain is equal to the number of evaluations of the hash function required to create the hash chain. For instance, a hash chain with values $x_0, \ldots, x_n$ is said to have length n. In this paper, we will use the hash chain as a primitive for a user to control the precision of the location information disclosed while still ensuring that the location proof remains valid using the approach described in [LMP11]. In the following, we describe the three mains algorithms used in this approach, mainly Hide, Reveal and Verify.

Without loss of generality, we assume that a location data *pos* can be represented by n location parameters, $pos = \{x_0, \ldots, x_n\}$. In practice, a well-formed GPS location data has a length $n = 3$ and a location $pos = (lat, long, time)$, in which *lat* is the (normalized) latitude, *long* is the longitude and *time* is the actual time. We assume that each location parameter x_i is a natural number and that it consists (possibly after padding with zeroes) of d digits. For instance for $1 \leq i \leq n$, we have $x_i = {x_i}^{d-1}{x_i}^{d-2} \ldots {x_i}^0$, where ${x_i}^{d-1}$ is the most significant digit of x_i and ${x_i}^0$ the least significant one. The accuracy of the parameter x_i can be controlled by hiding its rightmost digits. Let $pos|_p$ (for $d-1 \leq p \leq 0$) denotes the location *pos* of which the p least significant digits of each of the parameters are blinded. Let h be a cryptographic hash function satisfying pre-image resistance, second pre-image resistance and collision resistance. To encode a location data *pos* using the procedure Hide, Bob has to encode each location parameter x_i in *pos*. Indeed, for each

$x_i, (1 \leq i \leq n)$ in pos, Bob computes a hash chain $K_{x_i} = {K_{x_i}}^d, {K_{x_i}}^{d-1}, \ldots, {K_{x_i}}^0$ in which ${K_{x_i}}^0$ is a randomly chosen seed and ${K_{x_i}}^{j+1} = h({x_i}^j, {K_{x_i}}^j)$ (for $d-1 \geq j \geq 0$). Thus, the next hash value in a chain contains the following most significant digits of the location parameter x_i. Finally, the encoded value of pos is the hash chain for the different location parameters $K_{pos} = (K_{x_1}, K_{x_2}, \ldots, K_{x_n})$. Here, Bob gives K_{pos} to Alice. Alice compute S=Sign(K_{pos}) and sends S to Bob. Now, Bob can reveal different granularity over the position pos to Oscar using the procedure Reveal. To do so, Bob first determines a precision p (for $d-1 \leq p \leq 0$) and then reveals the hash value of the chain associated with that precision, ${K_{pos}}^p$, together with the partially blinded location $pos|_p$ to Oscar. Then, during the Verify procedure, Oscar reconstructs the parts of the hash following ${K_{pos}}^p$ and compares the final value with the one contained in the signature S. Using this method, Oscar only learns and checks the $d-p$ most significant bits of the location parameters certified by Alice, while the least significant bits remain unknown to him.

To encode the temporal information, the prover format the current time into five values (*i.e.*, $\{x_1^{'}, \ldots, x_5^{'}\}$) that correspond to the time (hh:mm:ss), period of the day (morning, afternoon or night), day, month and year. Then, he also relies on a hash chain to encode the temporal information in the LPS. The uncertainty of the position revealed depends of the number of digits hidden by the function Hide has illustrated in Table 7.1. A GPS coordinate relies on seven digits for the precision, thus the prover can hide a maximum of six digits.

Number of hidden digits	1	2	3	4	5	6
Radius in meters	0,1	1,3	13	136	1 369	13 701

Table 7.1: Radius of uncertainty with respect to the hidden digits.

7.4 Overview of PROPS

In this section, we describe in more details location proof gathering phase and the location proof verification phase.

7.4.1 Location proof gathering

The proof gathering protocol between a prover and a witness (summarized in Figure 7.2) consists of the following four rounds. The gathering process starts with an initialization phase in which the prover P generates random values key and $rand_1$ and uses them to compute the following values :

$K_{pos} = \mathsf{hide}(pos, key)$ and $K_{time} = \mathsf{hide}(time, key)$ and $C_1 = [\mathsf{Commit}(S_U^i, rand_i), i = 0, \cdots, m.]$

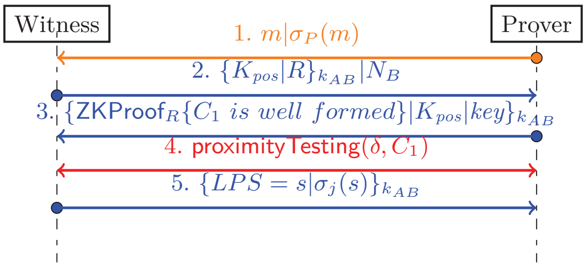
Witness
Prover
1. $m|\sigma_P(m)$
2. $\{K_{pos}|R\}_{k_{AB}}|N_B$
3. $\{\mathsf{ZKProof}_R\{C_1\ is\ well\ formed\}|K_{pos}|key\}_{k_{AB}}$
4. $\mathsf{proximityTesting}(\delta, C_1)$
5. $\{LPS = s|\sigma_j(s)\}_{k_{AB}}$

Figure 7.2: The location proof gathering phase.

The values K_{pos}, K_{time} represent the encryption of pos and $time$ under key, and C_1 is a serie of the homomorphic bit commitments on the each bit of the secret S_U. It also corresponds to the cryptographic secure pseudonym of the prover for this session.

The prover P then broadcasts a location proof request to his neighbouring users over an insecure (*i.e.* unencrypted and unauthenticated) channel:

$$(1) \quad P \rightarrow * : \{LPSReq = m|\sigma_{G,P}(m)\ ,\ m = pos|time|C_1|K_{pos}|K_{time}|N_A\}.$$

The request contains the following information: pos, the current position of the user and its encryption K_{pos}, $time$ the actual time and its encryption K_{time}, C_1 acting as a pseudonym, as well as N_A a share of the Diffie-Hellman key agreement and finally $\sigma_{G,P}(m)$ a group signature to authenticate the request.

Upon receiving the request Req, the witness W computes the following values: N_B a random share of a Diffie-Hellman key agreement, which is then combined with N_A in order to generate a session key k_{AB}. This session key k_{AB} will then be used to encrypt all subsequent communications between P and W by relying on a symmetric cryptosystem, thus ensuring the confidentiality and integrity of communications and R a fresh challenge for the zero-knowledge proof.

Afterwards, the witness W notifies the prover that he accepts to continue the generation process:

$$(2) \quad W \rightarrow P : \{LPSReply = \mathsf{Encrypt}((K_{pos}|R), k_{AB})|N_B\}.$$

The request $LPSReply$ contains the value of N_B in clear, the value K_{pos} and a challenge R encrypted as an acknowledgement to the previous phase. At the reception of $LPSReply$, P can reconstruct the session key k_{AB} using N_B and retrieves the challenge R of the zero-knowledge proof. P replies to W with the following zero-knowledge proof:

$$(3)\ P \rightarrow W : \{ZK = \mathsf{Encrypt}((\mathsf{ZKProof}(C_1\ wellformed)|K_{pos}|key), k_{AB})\}$$

The message ZK contains a zero-knowledge proof that the C_1 are valid commitments over the bits of the secret S_U certified by the CA. It also contains the value key to convince the witness that K_{pos} and K_{time} are valid encoding of pos and $time$.

$$\begin{aligned}\mathsf{ZKProof}\{(cert_{S_U}, S_U, rand_i) : C_1\} \leftarrow\ & \mathsf{ZKProof}\{(cert_{S_U}, S_U) : \\ & \mathsf{VerifyCommit}(C_1^i, S_U, rand_i) = 1\ \wedge \\ & \mathcal{CLS}.\mathsf{VerifySign}(S_U, cert_{S_U}, pk_{cert}) = 1\}.\end{aligned}$$

In order to ensure the freshness of the proof, the zero-knowledge proof will also depend on R, the nonce generated by the witness at the previous step. This is made possible by the use of zero-knowledge proofs based on CL-signatures.

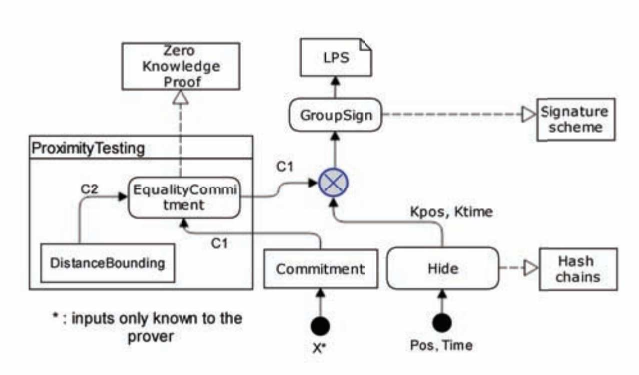

Figure 7.3: Overview of the gathering procedure.

Afterwards, P starts the proximity testing protocol with W.

$$(4)\quad P \leftrightarrows W : \{\mathsf{ProximityTesting}(\delta, C_1)\}$$

This protocol consists in a sequence of fast bit exchanges between the prover and the witness. More precisely, the witness selects a bit $b \in \{0, 1\}$ and sends it to the prover. Then, the prover replies with a response r. W records the time needed by the prover to produce the response to each of the challenges. These timings are used to estimate the proximity of the prover and this process is repeated several times in order to increase the robustness of the proximity testing protocol. If the timings are close enough to δ, then the $\mathsf{ProximityTesting}(\delta, C_1)$ procedure outputs accept, while otherwise it outputs reject. At this point, W is convinced that P is within proximity.

Finally, Finally, W creates a LPS $s = \{C_1|K_{pos}|K_{time}\}$ and a signature $\sigma_W(s) = \mathcal{UGS}.\mathsf{GroupSign}(s, gsk_W)$. P receives from W:

$$(5)\quad W \rightarrow P : \{\mathsf{Encrypt}(LPS, k_{AB})\}, LPS = s|\sigma_W(s)$$

Afterwards, the prover locally stores the LPSs collected from surrounding witness, the associated variables (the encoding key key) and the current spatio-temporal context ($time$ and pos).

After collecting at least k different shares coming from k unique witnesses, the prover P can aggregate them into a location proof that we denote thereafter as $certificate$.

7.4.2 Location proof verification

The proof verification phase described in Figure 7.4 enables a prover to convince a verifier that he was at a specific location at a particular moment in time. The verification process requires a proof of ownership by the current prover and also a check from the verifier that the location proof contains LPSs from different witnesses. If this last verification fails, then the verifier will forward the evidences of the fraudulent LPS to the AL who has the capacity to reveal the identity of the cheater using the opening key ok.

Without loss of generality, consider that the prover has collected k LPSs before sending a LP verification request to a verifier. A LP is generated as follows:

$$P \rightarrow V : LP = s|\sigma_{W_1}(s)|\ldots|\sigma_{W_k}(s)|aux_{pos}|L_{pos}|aux_{time}|L_{time}$$

in which L_{pos} (*resp.* L_{time}) represents the granularity of the position (respectively the time) to be revealed. The values aux_{pos} and aux_{time} are used to check that these granularities are conformed with the values K_{pos} and K_{time} of LP. These values are computed using the function Reveal (*cf.* Section 7.3.5).

Once he has received the LP , the verifier V runs sequentially the following verifications steps.

1. First, V checks for each LPS $\sigma_i(s)$ in $certificate$ whether $\mathcal{UGS}.\mathsf{GroupVerif}(s, \sigma_i(s), gpk)$ returns accept. More precisely, the function $\mathcal{UGS}.\mathsf{GroupVerif}(s, \sigma_{W_i}(s), gpk)$ verifies the validity of the group signature of each of the k LPSs $\sigma_i(s)$:

$$\bigwedge_{i=1}^{k}(\mathcal{UGS}.\mathsf{GroupVerif}(s, \sigma_{W_i}(s), gpk)) = \mathsf{true}.$$

2. Then, the verifier validates the uniqueness of the LPS in LP by verifying that *all these LPSs* have been generated by different witnesses:

$$\bigwedge_{i=1}^{k}\bigwedge_{j=i+1}^{k}(\mathcal{UGS}.\mathsf{Detect}(s, \sigma_{W_i}(s), \sigma_j(s))) = \mathsf{false}.$$

3. Next, V anonymously authenticates the prover P as the legal owner of LP by running a zero-knowledge proof protocol with him. At the end of this step, V is convinced that P knows the secret S_U used to generate the commitments C_1 and that this secret has been certified by the CA.

4. Finally, V uses the K_{pos} and K_{time} contained within LP to evaluate the validity of the spatio-temporal information (L_{pos}, L_{time}) claimed by the prover:

$$\mathsf{Check}(K_{pos}, aux_{pos}, L_{pos}) \wedge \mathsf{Check}(K_{time}, aux_{time}, L_{time}).$$

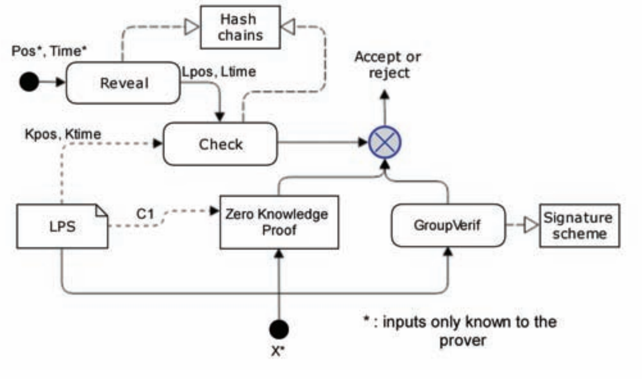

Figure 7.4: Overview of the verification procedure.

7.5 Protocol analysis

In this section, we analyze how Props fulfils the security and privacy properties described in Chapter 4.

7.5.1 Privacy

Anonymity and unlinkability of prover and witnesses. Due to the use of commitments and zero-knowledge proofs in Props, users can remain anonymous in the system as long as they behave honestly. Furthermore, the prover creates periodically nonces to generate the commitments C_1. Therefore, due to the hiding property of the underlying commitment scheme, these commitments do not disclose any information that can be used to trace back to the identity of the prover. In addition, the anonymity and unlinkability of witnesses are ensured by the use of group signature. Finally, the use of a zero-knowledge proof enables the prover to anonymously authenticate himself to the verifier as the owner of a LP.

Witness location privacy. When establishing a LP, a witness never discloses his exact position but rather checks that the position claimed by the prover is in the vicinity using the proximity testing protocol. Therefore, a local eavesdropper can only infer that the witness is in the proximity of the prover but does not learn his exact position.

Prover location privacy. The location information is first encoded into the hash chains before being endorsed by the witness. This information cannot be modified later by the prover and it does not appear in clear in the LP.

Location sovereignty. Within Props, the LPs gathered by a user are saved locally on his device in contrast with other schemes in which the proofs are stored and controlled by remote servers. Finally, due to the use of hash chains the prover can decide the granularity of the information he wants to disclose (*cf.* Section 7.3.5).

7.5.2 Security

Correctness. The correctness property is trivial. Once a LPS is received by the prover, he can verify that the spatio-temporal information contained within it is valid. The *spatial and temporal soundness* are ensured because revealing a geolocated context that does not match the one contained in the LP will be detected during the Step 4 of the verification process. Thus, a malicious prover cannot alter the integrity of a LPS and fool the verifier by claiming a different location than the one contained in the LPS. In the following, we give more details about how PROPS ensures the spatial soundness property (*cf.* Section 2.7.2) by proving its resistance to the distance fraud and mafia fraud.
Resistance to distance fraud. In a distance fraud, a malicious prover tries to convince an honest witness that he is closer than in reality. By assumption in PROPS, the distance fraud is prevented by the use of the $\mathsf{ProximityTesting}(\delta, C_1)$ protocol.
Resistance to mafia fraud. Let P be an honest prover located at position L_p and W an honest witness located at position L_w such that $\mathsf{dist}(L_p, L_w) > \delta$ with $dist(.,.)$ the euclidean distance. Consider $\bar{W}$ and $\bar{P}$, which are two different colluding users or the same malicious user playing two different roles. In the mafia fraud, the objective of an adversary is to replay a session that involved a honest prover P to fool W and make him believe that P is closer than he really is. Thus, two sessions need to be run, the first one involving $\bar{W}$ and P with the objective to get commitments from P followed by a second one performed between $\bar{P}$ and W in which the commitments of the first protocol are replayed in order to obtain a location share on behalf of P. In the second session $\bar{P}$ will need to compute a fresh zero-knowledge proof using R from W to prove knowledge of the identity S_U of P, which is impossible without the knowledge of S_U.

Proof of ownership and non-transferability. During the verification phase, the verifier checks that the current prover is effectively the legitimate owner of the LP by running a zero-knowledge protocol over the pseudonym C_1 included in the proof. Therefore, without the knowledge of the secret S_U involved in the creation of these commitments, it is impossible to claim the ownership of that location proof, which is also the case for a malicious verifier that wants to use a location proof that he has observed from an honest prover.
Within PROPS, the non-transferability property is equivalent to the resistance to the collusion $P - P$. The resilience to the collusion $P - P$ follows directly from the resilience to the *terrorist fraud* of the distance-boudning protocol we have used to implement the $\mathsf{ProximityTestting}$ tool.

Unforgeability. The unforgeability is ensured partially by the uniqueness property provided by unique group signature, which prevents the adversary controlling a collusion of m malicious users to gather enough LPSs as long as the size of the collusion is less than the number of shares needed (*i.e.*, $k > m$).
Resistance to distance hijacking. In a distance hijacking attack, a malicious user M tries to hijack the gathering session of an honest prover P. More precisely, M waits until P

has successfully proved that he is in the vicinity of an honest witness W and then hijacks P's session to collect its LPS. However in PROPS, a witness verifies that the entity who ran $\mathsf{ProximityTesting}(\delta, C_1)$, is the same as the one that computes the commitments C_1. Therefore, such an attack will be detected by W and the gathering process will be aborted before the malicious prover receives the LPS, thus avoiding the possibility of hijacking the session.

7.6 Implementation

In this section, we briefly report on the current proof-of-concept implementation of PROPS. Our main objective is to demonstrate that the architecture of PROPS can be implemented with currently available technology. Our implementation relies on Idemix[1] 2.3.4, a Java library containing advanced cryptographic primitives such as CL-signatures, commitment schemes and zero-knowledge proofs (*cf.* Section 7.3). Section 7.6.1 describes the test bench, while Section 7.6.2 gives the result of our experimentation.

7.6.1 Overview of the implementation

We implemented a Java prototype client application on Android. Our experiments are carried out on two Android devices: (1) a Samsung Galaxy Note 2 equipped with a Quad Core 1.6 GHz processor, 2GB of RAM, a GPS and running Android OS 4.1.2, acting as a prover, and (2) a Google Nexus 7 (2012 version) equipped with a Nvidia Tegra processor and 1 GB of RAM acting as a witness. The measurements that we report for each phase have been computed by averaging over 10 independent trials. In our testbed, we illustrate a scenario in which the witness listens to the network until he receives a location proof request from a nearby prover. As a result, the witness decides whether he accepts to serve the request or not. If the request is accepted, the witness sends an acknowledgement back to the prover. This process is denoted as the *initialization phase* in Figure 7.5. After the initialization phase, the witness checks that the prover possesses a valid credential from the CA on the value that is contained in the commitment (*authentication phase*), otherwise a cheating prover could encrypt gibberish or someone else's identity. Afterwards, the prover and the witness executes the *proof creation phase.* Finally, we also test the *proof sending phase* in which the witness sends the location share to the prover. During our experiments, we have measured the computational time (also an indicator of power consumption) and storage that are needed to run our current implementation PROPS on real smartphone devices.

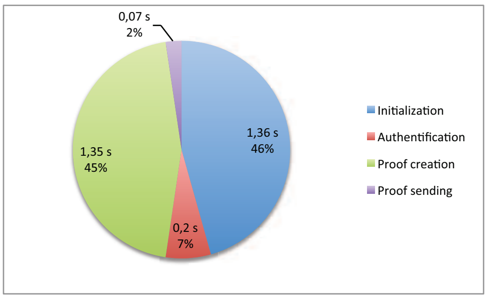

Figure 7.5: Cost of the different phases of the proof generation for the witness.

7.6.2 Experimental evaluation

From a memory point of view, each LPS has a size of 3444 bytes, which is higher compared to those of previous works. However, PROPS provides stronger privacy and security guarantees. The running time of the whole gathering phase conducted by a witness is on average of 2.98 ± 0.02 seconds.

For the verification phase, we have designed a realistic scenario in which a prover wants to show a LP to a remote LBS server. The server we used run on an Intel i5-2435M dual-core processor at 2.4 Ghz with 4 GB of 1333 Mhz DDR3 SDRAM running OSX 10.8.3. The average time needed to verify one LPS is around 0.66 ± 0.03 seconds for the verifier.

Unique group signature in the dynamic setting has been proven efficiently realizable [FZ12] under the CCA-anonymity assumption using any CCA-secure signature scheme (*i.e.*secure against adaptive chosen message attack) and Groth-Sahai proofs systems. In practice uniquely sign a message is equivalent to create three signatures and two Groth-Sahai proofs. Unique group signature in the dynamic setting verifies CPA-anonymity(*i.e.* the adversary cannot query the opening oracle. It is assumed that Opener is highly trusted and adversary cannot access it as long as it is honest) and CCA-anonymity. Our implementation of unique group signatures relies on the concept of domain pseudonym offered by the Idemix library. In a nutshell, a domain pseudonym can be used to link all the group signatures performed by a user within the same domain. Within the context of PROPS, we set the domain to the value $s = \{C_1|K_{pos}|K_{time}\}$ included in the location proof, thus allowing a verifier to check the uniqueness of the location proof share.

Regarding our proximity-testing black box, current implementation of distance bounding (*e.g.*, Rasmussen and Capkun [Rv10]) are hardware proof-of-concepts, thus they are not mature enough to be used on mobile devices like it is the case in PROPS. Nonetheless, one distance bounding protocol was really suitable for our needs in term of privacy (anonymity and unlinkability of the prover and his witnesses) but have been recently proved insecure against terrorist fraud [BBM+12]. From these facts, we have decided to abstract this distance bounding as a black-box while designing our protocol. In fact, the protocol is designed in such a way that it can be rearrange when needed. In practice, the lack of

[1] http://www.zurich.ibm.com/security/idemix/

implementation of distance bounding will surely endanger the security aspect (resistance to distance fraud, mafia fraud and terrorist fraud) of the protocol but not the privacy principles that we highlight in Chapter 4.

7.7 Conclusion

In this chapter, we introduced PROPS, a novel privacy-preserving location proof system based on a collaborative architecture. The main strengths of this location proof system are the following: (1) the LP collected by a prover are under his control and does not reveal any information about his identity, (2) the prover has the ability to remain anonymous even when presenting a proof to a verifier, (3) the privacy of users is preserved with respect to a man-in-the-middle adversary, (4) a verifier can detect abuses of a malicious user that tries to produce fake LPs, and finally (5) the prover can selective disclose the spatial and temporal information to the granularity of his choice. We also demonstrate the security of PROPS to standard attacks such as *collusion $P - P$/terrorist fraud*, *distance fraud*, *mafia fraud* and *replay attack*. In the future, we would like to extend PROPS to deal with the collusion $W - P$ in which a witness systematically reports false LPS for a colluder even though one or both of them are not at the location claimed in the LPS. Indeed, unless trusted infrastructures are deployed at each possible location, it seems quite difficult to detect that a particular LPS is a result of such a collusion [TCB12, WZP$^+$, ZC11]. A line of research that we would like to pursue in the future is the use of anonymous peer-to-peer reputation system as a countermeasure to frauds in this mobile environment. Finally, another research avenue is the design of a secure multiparty computation version of the protocol involving a joint interaction with the prover and multiple witnesses rather than relying on pairwise interactions between the prover and each witness.

Chapter 8

Conclusion and future work

8.1 Conclusion

Over the last four decades, mobile phones have become more than just a way to make calls. In fact, they are now “smart” enough to provide us real time assistance thanks to the existence of location-based services. Location-based services (LBS) are a class of services exploiting location data to deliver personalized features to its users. Examples of location-based services include the navigation systems found in many new cars, traffic advisories, roadside assistance and etc.

LBS help saving time in our daily tasks and become more productive in a world constantly moving. For example, a user looking for a gas station may just ask for it and the system provides him with the location and the directions to follow to get to his destination.

Actually, transmitting data regarding a user’s location to LBS causes two main problems. The first one is the lack of mechanisms for the LBS to check for the veracity of the data it receives from the user (*i.e.*, is the user attempting to access the service really present at the location he declares?). The second one is the potential other uses of data transmitted to the LBS because users are increasingly worrying about privacy and do not want governments to learn their whereabouts, stalkers to spy on them or even a spouse to monitor their movements.

In this thesis we have designed secure and privacy-enhanced protocols for authenticating distance and location claims for mobile devices. The systems we have designed in this work provide an answer to the location verification problem while embedding security mechanisms and privacy protection features. This work extends previously existing work by combining location proof system with distance-bounding protocols.

The first contribution of this thesis is the proposal of a distance-bounding protocol, as described in Chapter 6 that we named VSSDB. As introduced in Chapter 5, distance-bounding refers to a two-parties protocol that allows one entity called the verifier to authenticate another entity called the prover while at the same time ensuring that the distance separating them does not exceed a predefined threshold. Distance-bounding is traditionally ran in three phases which are the initialization, bit-exchange and verification phases. During the initialization phase, the prover and verifier initialize registers for the ongoing session. The bit exchange phase is a series of challenge-response sub-protocol called round. Each round of the bit-exchange phase is initiated by the verifier, usually by sending a challenge to the prover. Then, each challenge received by the prover is combined with the session registers to determine a response bit that is sent back to the verifier as soon as possible. Finally, during the verification phase, the verifier checks that the response for each round are related the session registers and that the time between sending a challenge and receiving the response from the prover does not exceed a predefined threshold.

VSSDB differs from the other distance-bounding protocols found in the literature because it does not need the prover and the verifier to share a secret key to create the session register. In fact, each prover is initiated with a public/private key pair. Then, the prover is authenticated by the verifier using the public key. This feature removes the verifier constraints to maintain huge database containing valid pseudonyms in the system then avoiding leakage of the private keys when a verifier gets corrupted by attackers. Therefore our system is more secure than state-of-the-art protocols.

Another novelty of our protocol is the use of modes, which are communicated during the initialization phase, and complement the challenges sent during the time-critical rounds. More precisely, the responses answered by the prover during one of these rounds depend both on the mode of this round and the challenge. As we demonstrated, this feature improves the protocol's security, while preserving the lightweightness of calculations performed during the time-critical rounds. We design two response functions for the VSSDB protocol. One that uses two responses modes and another which uses four. The two modes response function is well suited for environments with few privacy requirements because the private key of users need to be generated and checked by a central authority to guarantee their security. This condition is ruled out by the four modes responses functions. In addition, in contrast to other distance-bounding protocols that use secret-sharing technique to hide the secret known to the prover within the responses bits, we rely on the use of three shares rather than two, which allows for more richness in the responses while better hiding the secret from man-in-the-middle attacker.

We have also defined in this thesis a notion of terrorist-fraud resistance called KeyTF-security. Essentially, KeyTF-security is a relaxation of GameTF-security [FO13b]. While our VSSDB protocol is KeyTF-secure, we also describe a extended version to gain GameTF-security.

We analyse the security features of the VSSDB protocol in the framework of Dürholz, Fischlin, Kasper and Onete [DFKO11b]. The results of this formal analysis indicate that the security properties stated for the protocol are upheld.

The second contribution of this thesis is the design of a privacy-enhanced location proof system for use with mobile devices. A location proof system denotes a system allowing its users to prove their location at a particular time to remote entities called verifiers.

In Chapter 3 we have addressed state-of-the-art systems and given a detailed description of how they work. This study has revealed that there exists two main families of location proof systems. The first one includes systems where users collect location proof by interacting with dedicated access-Point disseminated over the localization area. We call such an approach of implementing location proof system, the bipartite gathering approach. Secondly, we investigate systems in which users collect location proof by collaborating together. We call this latter approach the collaborative gathering approach.

Then within Chapter 4, we have investigated the features offered by the state-of-the-art location proofs system in terms of security and privacy. This analysis allowed us to investigate the possibility of vulnerabilities in the design of the protocols and how we can remedy to them. In general, many of the previous works only considered location proof as just a signature of location data attesting the presence of a user at a location at a particular time. In fact, a location proof system must include countermeasures against various frauds and attacks inspired from the domain distance-bounding protocol and cryptography. We enumerate those threats and give scenarios to justify why they must be taken into account in the design of location proof systems. We encapsulate all our recommendations into a unified framework compatible with previous works in the domain. Our framework can be used at any phase during future development of location proof system. It includes an adversary model that differs from the others as we give more possibilities to verifier in attacking the system. In fact, the verifier's role can be played by any party without requiring the prover to trust him. Additionally, the framework models a location proof system as a tuple of algorithms ($\mathsf{Setup}, \mathsf{URegister}, \mathsf{VRegister}, \mathsf{LTestify}, \mathsf{LProve}, \mathsf{Vf}$) and enumerates the threats to be considered while designing each part of a location proof system. Because privacy is of central importance to mobile users, we also identify the privacy requirements for each of these algorithms. In this work, we defined privacy in terms of anonymity of the users within the system and the unlinkability of their actions. Finally, we give a complete analysis of the state-of-the-art protocols within the framework we proposed.

From the lessons learned in previous chapters, we proposed a location proof system that we named PROPS (See Chapter 7) to solve the identified weaknesses of related work. PROPS employs an ad-hoc approach to the problem of location proof, enabling neighbouring devices to provide a certificate which attests of a specific device's presence in a claimed area. We employed distance-bounding to establish a location proof regarding the presence of a device called the prover in a particular area, according to a neighbouring device called the witnesses. In our scheme, the proximity of the prover device to a witness is limited to a single hop communication range only. Anything above this is detected by the underlying DB protocol and tagged as a proxy exchange, essentially preventing relay attacks. By employing DB to protect the location proof gathering process, the soundness of the location proof is guaranteed, as tampering with the distance will be detected and the gathering session aborted.

Additionally, by protecting the confidential information pertaining to the participants during a session of the protocol, it is more difficult for an adversary to identify who has participated.

Malicious participants are prevented from linking the identity of other participants to running sessions through the use of changing pseudonyms. The protocol protects against external observers by encrypting the sensitive data transmitted during the slow phase of the protocol. Due to the use of distance-bounding, it is obvious that local observers could deduce the position of devices participating in a gathering session based on traffic and message content analysis. However, they are not capable of discovering the identity of the participants or any information to uniquely identify them. To better ascertain the properties of our protocol, we analyse its properties according to the framework we have developed in Chapter 4 and then compare our work to previous works. This latter comparison shows that our protocol enables better security and privacy features than previous proposals.

8.2 Perspectives

As we introduced in Chapter 2 the location verification problem has become of significant importance with recent research turning towards ad-hoc systems. In this dissertation, we mainly focus on the design of security protocols for authenticating distance and location claims for mobile devices with an effort to move away from infrastructure dependence. While the work presented in this dissertation is comprehensive, there remain a number of avenues for future research.

A first research direction of this work would be to define verifier-side mechanisms to set trust score to a prover according to the location proof he discloses. I would be interesting to extend PROPS to deal with the collusion $W - P$ in which a witness systematically reports false location proof share for a colluder even though one or both of them are not at the location claimed in the LPS. At first glance, unless some sort of trusted computing is deployed at each possible location, it seems quite difficult for the verifier to detect that a particular location proof shares is a result of such a collusion [TCB12, WZP$^+$, ZC11]. Therefore, a line of research that we would like to pursue in the future is the use of anonymous peer-to-peer reputation system as a countermeasure to the above mentioned fraud in this particular setting.

Another research avenue would be the design of a secure multiparty computation version of PROPS involving a joint interaction with the prover and multiple witnesses rather than relying on pairwise interactions between the prover and each witness.

We also think that the popularity of distance-bounding protocols will become more and more important because of the importance of relay attacks within traditional authentication systems. A future research direction in this domain would be to obtain privacy for VSSDB, of which the asymmetric nature of the prover secret key is a necessary first step, as well as investigating the possibility of using other secret-sharing techniques to obtain better distance- and terrorist-fraud resistance.

.

Bibliography

[95412] *New draft European data protection regime to apply also to all US companies processing data of European residents*, 2012. http://mlawgroup.de/news/publications/detail.php?we_objectID=227.

[ABK+11] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardas, Cédric Lauradoux, and Benjamin Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 19(2):289–317, 2011.

[ADG+12] Christian Artigues, Yves Deswarte, Jérémie Guiochet, Marie-José Huguet, Marc-Olivier Killijian, David Powell, Matthieu Roy, Christophe Bidan, Nicolas Prigent, Emmanuelle Anceaume, et al. Amores: an architecture for mobiquitous resilient systems. In *Proceedings of the 1st European Workshop on AppRoaches to MObiquiTous Resilience*, page 7. ACM, 2012.

[AK12] Mads Schaarup Andersen and Mikkel Baun Kjærgaard. Towards a new classification of location privacy methods in pervasive computing. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 150–161. Springer, 2012.

[ALM11] Gildas Avoine, Cédric Lauradoux, and Benjamin Martin. How secret-sharing can defeat terrorist fraud. In *Fourth ACM Conference on Wireless Network Security, WISEC 2011*, pages 145–156, Hamburg, Germany, June 2011. ACM.

[APR+99] Ward Andy, Steggles Pete, Curwen Rupert, Webster Paul, Addlesee Mike, Newman Joe, Osborn Paul, and Steve Hodges. *The Active Bat indoor positioning system*, 1999. http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/.

[AT09a] Gildas Avoine and Aslan Tchamkerten. An efficient distance bounding rfid authentication protocol: balancing false-acceptance rate and memory requirement. In *Information Security*, pages 250–261. Springer, 2009.

[AT09b] Gildas Avoine and Aslan Tchamkerten. An efficient distance bounding RFID authentication protocol: Balancing false-acceptance rate and memory requirement. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio Ardagna, editors, *Information Security*, volume 5735 of *Lecture Notes in Computer Science*, pages 250–261. Springer Berlin / Heidelberg, 2009.

[B$^+$] George Robert Blakley et al. Safeguarding cryptographic keys.

[BB05a] Laurent Bussard and Walid Bagga. Distance-Bounding Proof of Knowledge to Avoid Real-Time Attacks. In *International Conference on Information Security - SEC 2005*, pages 223–238, Chiba, Japan, June 2005. Springer Verlag.

[BB05b] Laurent Bussard and Walid Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In *Security and Privacy in the Age of Ubiquitous Computing*. 2005.

[BBD$^+$91] Samy Bengio, Gilles Brassard, Yvo G Desmedt, Claude Goutier, and Jean-Jacques Quisquater. Secure implementation of identification systems. *Journal of Cryptology*, pages 175–183, 1991.

[BBM$^+$12] Asli Bay, Ioana Boureanu, Aikaterini Mitrokotsa, Iosif Spulber, and Serge Vaudenay. The Bussard-Bagga and Other Distance-Bounding Protocols under Attacks. In *Information Security and Cryptology - 8th International Conference, Inscrypt 2012*, Lecture Notes in Computer Science 7763, pages 371–391, Beijing, China, November 2012. Springer.

[BC93] Stefan Brands and David Chaum. Distance-Bounding Protocols. In *Advances in Cryptology – EUROCRYPT'93*, Lecture Notes in Computer Science 765, pages 344–359, Lofthus, Norway, 1993. Springer-Verlag.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, pages 156–189, 1988.

[BCKM04] Ralf Bill, Clemens Cap, Martin Kofahl, and Thomas Mundt. Indoor and outdoor positioning in mobile environments—a review and some investigations on wlan-positioning. *Geographic Information Sciences*, 10(2), 2004.

[BCL06] Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Security Protocols*. Springer Berlin Heidelberg, 2006.

[BMV13a] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Secure and lightweight distance-bounding. In *Lightweight Cryptography for Security and Privacy*, pages 97–113. Springer, 2013.

[BMV13b] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Towards secure distance bounding. *the 20th anniversary annual Fast Software Encryption (FSE 2013)*, 2013.

[BS03] Alastair R Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, (1):46–55, 2003.

[Cav11] Ann Cavoukian. Privacy by design: Origins, meaning, and prospects. *Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards: Aspects and Standards*, page 170, 2011.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *Symposium on Foundations of Computer Science - FOCS 1985*, pages 383–395, Portland, OR, USA, October 1985. IEEE Computer Society.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks*, Lecture Notes in Computer Science, 2003.

[Con76] John Horton Conway. *On numbers and games*, volume 6. IMA, 1976.

[Cot11] Caitlin D Cottrill. Location privacy: Who protects? *URISA Journal-Urban and Regional InformationSystems Association*, 23(2):49, 2011.

[CP12] Bogdan Carbunar and Rahul Potharaju. You unlocked the Mt. Everest badge on foursquare! Countering location fraud in Geosocial Networks. In *IEEE International Conference on Mobile Ad-Hoc and Sensor, MASS*, pages 182–190, Las Vegas, NV, USA, October 2012. IEEE Computer Society.

[CRSC12] Cas Cremers, Kasper Bonne Rasmussen, Benedikt Schmidt, and Srdjan Capkun. Distance hijacking attacks on distance bounding protocols. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 113–127. IEEE, 2012.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology.* 1997.

[CTM07] Yingying Chen, Wade Trappe, and Richard P Martin. Attack detection in wireless localization. In *INFOCOM 2007. 26th IEEE International*

Conference on Computer Communications. IEEE, pages 1964–1972. IEEE, 2007.

[CvH91] David Chaum and Eugene van Heyst. Group signatures. In *EUROCRYPT*. LNCS 547, 1991.

[DCF12] Benjamin Davis, Hao Chen, and Matthew Franklin. Privacy-preserving alibi systems. In *7th ACM Symposium on Information, Computer and Communications Security, ASIACCS*, Seoul, South Korea, 2012.

[Dem03] Mike Dempsey. *Indoor Positioning Systems in Healthcare*, 2003. `http://www.cimit.org/pubs/ipsinhealthcare.pdf`.

[Des88] Y. Desmedt. Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them. In *6th Worldwide Congress on Computer and Communications Security and Protection - SecuriCom '88*, 1988.

[DF02] Ivan Damgard and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *Advances in Cryptology - ASIACRYPT 2002*. Springer Berlin Heidelberg, 2002.

[DFKO11a] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A formal approach to distance bounding RFID protocols. In *Proc. of ISC'11*, volume 7001 of *LNCS*, pages 47–62. Springer-Verlag, 2011.

[DFKO11b] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A formal approach to distance-bounding rfid protocols. In *Information Security*, volume 7001 of *Lecture Notes in Computer Science*, pages 47–62. Springer Berlin Heidelberg, 2011.

[DR01] Goran M Djuknic and Robert E Richton. Geolocation and assisted gps. *Computer*, 34(2):123–125, 2001.

[FDv11] Aurélien Francillon, Boris Danev, and Srdjan Čapkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. In *Network and Distributed System Security Symposium, NDSS 2011*, San Diego, CA, USA, February 2011. The Internet Society.

[Fel87a] Paul Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *Symposium on Foundations of Computer Science - FOCS 1987*, pages 427–437, Los Angeles, CA, USA, October 1987. IEEE Computer Society.

[Fel87b] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Foundations of Computer Science, 28th Annual Symposium on*, pages 427–438. IEEE, 1987.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.

[FHMM10] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In *International conference on Radio frequency identification: security and privacy issues - RFIDSec'10*, pages 35–49, Istanbul, Turkey, 2010. Springer-Verlag.

[FNI13] Zahid Farid, Rosdiadee Nordin, and Mahamod Ismail. Recent advances in wireless indoor localization techniques and system. *Journal of Computer Networks and Communications*, 2013, 2013.

[FO12] Marc Fischlin and Cristina Onete. Provably secure distance-bounding: an analysis of prominent protocols. *IACR Cryptology ePrint Archive*, 2012:128, 2012.

[FO13a] Marc Fischlin and Cristina Onete. Subtle kinks in distance-bounding: an analysis of prominent protocols. In *Proc. WISec'13*, pages 195–206. ACM Press, 2013.

[FO13b] Marc Fischlin and Cristina Onete. Terrorism in distance bounding: Modeling terrorist fraud resistance. In *Proceedings of ACNS'13*, volume 7954 of *LNCS*, pages 414–431. Springer-Verlag, 2013.

[Fra] Al Franken. *The Location Privacy Protection Act of 2014*. `http://www.franken.senate.gov/files/documents/140327Locationprivacy.pdf`.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*. LNCS 263, 1987.

[FZ12] Matthew Franklin and Haibin Zhang. Unique group signatures. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS*, pages 643–660. LNCS 7459, 2012.

[GG09] Michelle Graham and David Gray. Protecting privacy and securing the gathering of location proofs - the secure location verification proof gathering protocol. *LNICST*, 17, 2009.

[GKaT12] S. Gambs, Marc-Olivier Killijian, and M. Roy andM. Traore. Locanyms: Towards privacy-preserving location-based services. In *1st European Workshop on AppRoaches to MObiquiTous Resilience (ARMOR'2012)*, 2012.

[GKC$^+$13] Sébastien Gambs, Marc-Olivier Killijian, Miguel Nunez Del Prado Cortez, Moussa Traoré, et al. Towards a recommender system for bush taxis. In *3rd Conference on the Analysis of Mobile Phone Datasets (NetMob'13)*, 2013.

[GKdPC10] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show me how you move and i will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, pages 34–41, New York, NY, USA, 2010. ACM.

[GKL$^+$] Sébastien Gambs, Marc-Olivier Killijian, Cédric Lauradoux, Cristina Onete, Matthieu Roy, and Moussa Traoré. Vssdb: A verifiable secret-sharing and distance-bounding protocol. In *International Conference on Cryptography and Information security (BalkanCryptSec'14)*.

[GKRT14] Sébastien Gambs, Marc-Olivier Killijian, Matthieu Roy, and Moussa Traoré. Props: A privacy-preserving location proof system. In *2014 IEEE 33rd International Symposium on Reliable Distributed Systems (SRDS)*, pages 1–10. IEEE, 2014.

[GLN09] Yanying Gu, A Lo, and I Niemegeers. A survey of indoor positioning systems for wireless personal networks. *Communications Surveys Tutorials, IEEE*, 11(1):13–32, First 2009.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *ACM STOC*, 1985.

[Gol98] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 1998.

[GOR14] Sébastien Gambs, Cristina Onete, and Jean-Marc Robert. Prover Anonymous and Deniable Distance-Bounding Authentication. In *Proceedings of ACM AsiaCCS'14, Accepted for publication.* ACM Press, 2014.

[GQ88] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *EUROCRYPT*. 1988.

[GS07] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007 155, 2007.

[Han12] Gerhard P Hancke. Distance-bounding for rfid: Effectiveness of 'terrorist fraud'in the presence of bit errors. In *RFID-Technologies and Applications*

(RFID-TA), 2012 IEEE International Conference on, pages 91–96. IEEE, 2012.

[HHP03] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 99–110. ACM, 2003.

[HK05a] Gerhard P. Hancke and Markus G. Kuhn. An RFID distance bounding protocol. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 67–73. IEEE, 2005.

[HK05b] G.P. Hancke and M.G. Kuhn. An rfid distance bounding protocol. In *Procs of SecureComm 2005*, pages 67 – 73, sept. 2005.

[HPALP09] Ville Honkavirta, Tommi Perälä, Simo Ali-Löytty, and Robert Piché. A comparative survey of wlan location fingerprinting methods. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*, pages 243–251. IEEE, 2009.

[HPO13] Jens Hermans, Roel Peeters, and Cristina Onete. Efficient, secure, private distance bounding without key updates. In *ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC'13*, pages 207–218, Budapest, Hungary, April 2013. ACM.

[HWK04] Qi He, Dapeng Wu, and Pradeep Khosla. The quest for personal control over mobile location privacy. *Communications Magazine, IEEE*, 42(5):130–136, May 2004.

[ISO] *ISO/IEC 29100 Directives.* `https://www.iso.org/obp/ui/#iso:std:iso-iec:29100:ed-1:v1:en`.

[KAK+09] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. The swiss-knife RFID distance bounding protocol. In *Information Security and Cryptology–ICISC 2008*, pages 98–115. Springer, 2009.

[LH10] Wanying Luo and Urs Hengartner. Proving your location without giving up your privacy. In *ACM HotMobile*, 2010.

[LMP11] Gabriele Lenzini, Sjouke Mauw, and Jun Pang. Selective location blinding using hash chains. In *Security Protocols Workshop*, LNCS 7114, 2011.

[Mar12] Konstantinos Markantonakis. Practical relay attack on contactless transactions by using nfc mobile phones. *Radio Frequency Identification System Security: RFIDsec*, 12:21, 2012.

[MP08] Jorge Munilla and Alberto Peinado. Security analysis of tu and piramuthu's protocol. In *New Technologies, Mobility and Security, 2008. NTMS'08.*, pages 1–5. IEEE, 2008.

[NXP11] NXP. MF1PLUSx0y1 Mainstream contactless smart card IC for fast and easy solution development. Technical report, 2011. Revision 3.2.

[oNC11] ACLU of Northern California. *Location-Based Services: Time for a Privacy check-in*, 2011. `http://aclunc-tech.org/files/lbs-privacy-checkin.pdf`.

[PC] DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT and OF THE COUNCIL. *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.* `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML`.

[PCB00] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43. ACM, 2000.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology-CRYPTO'91*, pages 129–140. Springer, 1992.

[PLHCvdLT09] Pedro Peris-Lopez, Julio C Hernandez-Castro, Jan CA van der Lubbe, and JME Tapiador. Shedding some light on rfid distance bounding protocols and terrorist attacks. Technical report, 2009.

[PVA00] Bahl Paramvir, Padmanabhan Venkata, N., and Balachandran Anand. Enhancements to the radar user location and tracking system. Technical report, Microsoft Reserach, University of California at San Diego, 2000.

[RMT$^+$02] Teemu Roos, Petri Myllymäki, Henry Tirri, Pauli Misikangas, and Juha Sievänen. A probabilistic approach to wlan user location estimation. *International Journal of Wireless Information Networks*, 9(3):155–164, 2002.

[RNTS07] Jason Reid, Juan M Gonzalez Nieto, Tee Tang, and Bouchra Senadji. Detecting relay attacks with timing-based protocols. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 204–213. ACM, 2007.

[RTv$^+$12] Aanjhan Ranganathan, Nils Ole Tippenhauer, Boris Škorić, Dave Singelée, and Srdjan Čapkun. Design and implementation of a terrorist-fraud resilient distance bounding system. In *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12)*, volume 7459 of *LNCS*, pages 415 – 432. Springer-Verlag, 2012.

[Rv10] Kasper Bonne Rasmussen and Srdjan Čapkun. Realization of rf distance bounding. In *Proceedings of the USENIX Security Symposium*, 2010.

[Sch94] Bruce Schneier. Applied cryptography–protocols, algorithms, and... 1994.

[Sha79] Adi Shamir. How to share a secret. *Communication of the ACM*, 22(11):612–613, 1979.

[She10] Sidney Shek. Next-generation location-based services for mobile devices. In *Leading Edge Forum, Computer Science Corporation*, pages 1–66, 2010.

[SW07] Avinash Srinivasan and Jie Wu. A survey on secure localization in wireless sensor networks. *Encyclopedia of Wireless and Mobile communications*, 2007.

[SW09] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *ACM HotMobile*, 2009.

[TCB12] Manoop Talasila, Reza Curtmola, and Cristian Borcea. Link: Location verification through immediate neighbors knowledge. *Springer, LNICST 73*, 2012.

[TP07] Yu-Ju Tu and Selwyn Piramuthu. Rfid distance bounding protocols. In *First International EURASIP Workshop on RFID Technology, Vienna, Austria (September 2007)*, 2007.

[TRPČ09] Nils Ole Tippenhauer, Kasper Bonne Rasmussen, Christina Pöpper, and Srdjan Čapkun. Attacks on public wlan-based positioning systems. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 29–40. ACM, 2009.

[Vau13] Serge Vaudenay. On modeling terrorist frauds – addressing collusion in distance bounding protocols. In *Proceedings of the 7th Conference on Provable Security (ProvSec'13)*, volume 8209 of *LNCS*, pages 1–20. Springer-Verlag, 2013.

[VJS+07] Hien Nguyen Van, Yunye Jin, Wee-Seng Soh, et al. Indoor localization using multiple wireless technologies. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on*, pages 1–8. IEEE, 2007.

[Wes68] Alan F Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[WF03] Brent Waters and Edward Felten. Secure, private proofs of location. Technical report, Department of Computer Science, Princeton University, Tech. Rep. TR-667-03, 2003.

[WHFaG92] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, January 1992.

[WZP+] Xinlei Oscar Wang, Jindan Zhu, Amit Pande, Arun Raghuramu, Prasant Mohapatra, Tarek Abdelzaher, and Raghu Ganti. Stamp: Ad hoc spatial-temporal provenance assurance for mobile users.

[YLAU11] Lei Yu, Mohamed Laaraiedh, Stéphane Avrillon, and Bernard Uguen. Fingerprinting localization based on neural networks and ultra-wideband signals. In *Signal Processing and Information Technology (ISSPIT), 2011 IEEE International Symposium on*, pages 184–189. IEEE, 2011.

[ZC11] Zhichao Zhu and Guohong Cao. Applaus: A privacy-preserving location proof updating system for location-based services. In *INFOCOM*, pages 1889–1897, 2011.