DEGREE PROJECT IN ELECTRICAL ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Pedestrian to Vehicle Communication: A Safe and Private Solution Proposal

PABLO SÁNCHEZ CARMONA

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Pedestrian to Vehicle Communication: A Safe and Private Solution Proposal

PABLO SÁNCHEZ CARMONA

Degree Programme in Electrical Engineering
Date: March 22, 2020
Supervisor: Mohammad Khodaei
Examiner: Panagiotis Papadimitratos
School of Electrical Engineering and Computer Science (EECS)
Swedish title: Fotgängare till fordonskommunikation. Ett säkert och privat lösningsförslag

Acknowledgements

This document is a summary of a journey that started about one year ago and, although it took longer than expected, it leaves me with very good feelings. After seven years, a stage of my life is ending and I would like to use these lines to thank all the people who have helped me during these years.

First to Panos for being understanding but always demanding, and for helping me to arrange all the ideas in my brain, allowing me to develop something that can really become a change in this world. I would also like to thank the NSS group and the School of Electrical Engineering and Computer Science for letting me use the necessary tools to carry out this project.

Tack vare KTH, för de senaste två åren har jag lärt mig nya sätt att arbeta och lära, och nu är jag mycket mer beredd att utveckla min framtid. Jag skulle aldrig glömma de två fantastiska åren i Stockholm.

També m'agradaria agrair a en Josep que m'hagi aguantat aquestes últimes setmanes i que m'hagi ajudat des de la distància. També agrair que en el seu moment m'ensenyés quelcom nou dins del meu sector que jo desconeixia, que m'ha aportat molt i de ben segur em serà de gran ajuda allà on faci servir els meus coneixements. A l'ETSETB i la UPC, gràcies per oferir-me una educació de qualitat sense haver d'anar molt lluny.

No me quiero olvidar todo lo que me ha aportado la ETSIIT en la UGR, dándome la base que hace que ahora todo resulte fácil, y enseñándome que un teleco es aquel que crea para ayudar y aportar. No tengo más que elogios a la universidad pública, que siempre ha estado a la altura de las mejores (o más). Que nunca nos quiten ese privilegio.

In a more personal sense, thanks to MAIN for allowing me to enjoy the greatest culture mix that I will have in my entire life. We have achieved something amazing, and I only hope that those who continue will have the same strength and same fondness that we had those who were from the beginning. I made true friends in the path, and I am very grateful for that.

Y por último, y por ello más importante, a mi familia y a mis amigos; que han estado desde el principio y aquellos, que aunque no llevan mucho, estoy seguro que han llegado para quedarse. A quién me soporta y ha hecho que estos fríos y oscuros meses sean increíbles, muchas gracias.

It may seem like the end, but this has not done anything but start.

Abstract

Vehicle-to-vehicle communication (V2V) has become one of the most rising technologies in recent years, and due to the implementation of 5G different solutions have begun to be designed that allow communication between vehicles and pedestrians (V2P). V2P communication is a challenge for two reasons: the incompatibility of technologies, as well as the need to have a real-time communication (crucial when sending and receiving road hazard messages). This document presents a solution for pedestrian-to-vehicle communication, detecting pedestrian risk situations using the mobile phone sensors and messages by the vehicles. An architecture and communication protocol is presented, to send warning messages with low delay in a secure and private way without harming efficiency. Two algorithms are used, one implemented in an Android application and another one in a server, with the purpose of detecting pedestrian misbehavior in the road and reducing the amount of direct messages between the nodes of the system.

Keywords — Android, Privacy, Secure communication, User safety, Pedestrian-to-vehicle.

Resumen

La comunicación entre vehículos (V2V) se ha convertido en una de las tecnologías más emergentes en los últimos años, y debido a la implementación de tecnologías 5G, la comunicación peatón-vehículo (V2P) es un tema actual a solucionar. La comunicación V2P supone un desafío para su implementación por dos razones principales: la incompatibilidad de tecnologías, así como la necesidad de tener una comunicación en tiempo real (crucial en situaciones en carretera). Este documento presenta una solución para la comunicación entre peatones y vehículos, detectando situaciones de riesgo creadas por peatones con los sensores del teléfono móvil y recibiendo mensajes por parte de los vehículos. Se presenta una arquitectura completa y un protocolo de comunicación para enviar mensajes con el menor retraso posible y de forma segura y privada, sin perder la eficiencia del sistema. Se utilizan dos algoritmos, uno implementado en una aplicación de Android y otro en un servidor, con el fin de detectar cuando los peatones crean una situación de peligro real y al mismo tiempo reducir la cantidad de mensajes diretos enviados entre los nodos del sistema.

Palabras clave — Android, Privacidad, Comunicación Segura, Seguridad del peatón, Comunicación peatón-vehículo.

Sammanfattning

Kommunikation mellan fordonen (V2V) har blivit en av de mest stigande teknikerna under de senaste åren, och på grund av implementeringen av 5G har olika lösningar börjat utformas som möjliggör kommunikation mellan fordon och fotgängare (V2P). V2P-kommunikation är en utmaning av två skäl: teknikens inkompatibilitet, liksom behovet av att ha en kommunikation i realtid (avgörande när du skickar och tar emot meddelanden om vägfara). Detta dokument presenterar en lösning för kommunikation mellan fotgängare och fordon, från upptäckten av fotgängarsrisksituationer på vägen använder mobiltelefonsensorer till fordonen mottagande meddelanden. Ett fullständigt arkitektur- och kommunikationsprotokoll presenteras för att skicka flera varningsmeddelanden med minsta möjliga fördröjning och på ett säkert och privat sätt, utan att förlora systemets effektivitet. Dessutom används två algoritmer, en implementerad i en Android-applikation och en annan på en server, i syfte att upptäcka uppförande av fotgängare på vägen och minska mängden direktmeddelanden mellan systemets noder.

Nyckelord — Android, Sekretess, Säker Kommunikation, Användarsäkerhet, Fotgängare-till-fordon.

Resum

La comunicació entre vehicles (V2V) s'ha convertit en una de les tecnologies més emergents en els últims anys, ia causa de la implementació de tecnologies 5G, la comunicació vianant-vehicle (V2P) és un tema actual a solucionar. La comunicació V2P suposa un desafiament per a la seva implementació per dues raons principals: la incompatibilitat de tecnologies, així com la necessitat de tenir una comunicació en temps real (crucial en situacions en carretera). Aquest document presenta una solució per a la comunicació entre vianants i vehicles, des de la detecció de situacions de risc creades per vianants a la carretera amb els sensors del telèfon mòbil, fins a la recepció d'aquests missatges per part dels vehicles. Es presenta una arquitectura completa i un protocol de comunicació per enviar missatges amb el menor retard possible i de forma segura i privada, sense perdre l'eficiència del sistema. A més, s'utilitzen dos algoritmes, un implementat en una aplicació d'Android i un altre a un servidor, per tal de detectar quan els vianants creen una situació de perill real i al mateix temps reduir la quantitat de missatges diretos enviats entre els nodes del sistema.

Paraules Clau — Android, Privadesa, Comunicació Segura, Seguretat del vianant, Comunicació vianant-vehicle.

List of Figures

List of Tables

Contents

Acronyms

5G/4G 5th Generation / 4th Generation.

AES Advanced Encryption Standard.

API Application Programming Interface.

ATT Attribute Protocol.

BLE Bluetooth Low Energy.

CBC Cipher Block Chaining.

CRL Certificate Revocation List.

DES Data Encryption Standard.

DoS Denial of Service.

DSRC Dedicated Short Range Communications.

DTLS Datagram Transport Layer Security.

ECDSA Elliptic Curve Digital Signature Algorithm.

GATT Generic Attribute Profile.

GPS Global Position System.

IDEA International Data Encryption Algorithm.

IIoT Industrial Internet of Things.

IoT Internet of Things.

IPSec Internet Protocol Security.

IRK Identity Resolving Key.

ISAKMP Internet Security Association and Key Management Protocol.

LED Ligth-Emitting Diode.

LOS Line-of-Sight.

M2M Machine-to-machine communication.

MAC Message Authentication Code.

NLOS Non Line-of-Sight.

OBU On-Board Unit.

PKI Public Key Infrastructure.

RSU RoadSide Unit.

SA Security Association.

SADB Security Association Data Base.

SI International System of Units.

SSL Secure Socket Layer.

TCP Transport Control Protocol.

TLS Transport Layer Security.

UDP User Datagram Protocol.

V2P Vehicle-to-pedestrian communication.

V2V Vehicle-to-vehicle communication.

V2X Vehicle-to-anything communication.

VANET Vehicular Ad-hoc Network.

VPKI Vehicular Public-Key Infrastructure.

VPN Virtual Private Network.

WAVE Wireless Access in Vehicluar Environments.

Chapter 1

Introduction

The advance towards a full-connected world includes two trends: V2X (vehicle to anything) communication and IoT (Internet of Things). Many universities and companies are working on the challenge of the communication of nodes between different networks, especially now with the deployment of 5G technologies. Talking about V2X technologies, most of the implementations are focused on communications between vehicles (V2V) and between vehicles and infrastructures (V2I - I2V). Most of the solutions look for an easy developments in terms of architecture and security: a single architecture without merging technologies, allowing to secure the network effectively. Vehicles communicate in order to provide road, transportation safety; but the problem at hand here arises when we think about pedestrians or light vehicles such as bicycles.

Two key factors need to be considered in vehicle-to-pedestrian (V2P) communication: the detection of dangerous situations, and the exchange of messages between cars and pedestrians. About the first issue, several solutions have been presented. For example [1], [2] and [3] focus on the use of cameras and sound detectors in mobile phones (assisted by other built-in sensors) in order to detect dangerous situations in the road. Nevertheless, sensitive information is processed, such as pictures and sounds, compromising the privacy of the user. Besides, those solutions only apply if the user is holding the phone, and they don't work with the phone in the pocket or in a bag. In [4] another solution is presented, updating the location of both vehicles and pedestrians and determining when a collision is going to happen, without analyzing another factors in the pedestrian behaviour.

The second issue is the most critical one, how the data exchange is going to be done by the nodes. The introduction of pedestrians introduces several issues that need to be taken into account, such as the technology incompatibility between the vehicles and pedestrians (hardware and communication protocols). The introduction of a new type of nodes - pedestrians behaviour is much different as cars - and the large number of these nodes in comparison to cars - in urban and suburban areas - are also important variables that should be overcome when designing solutions. Compelling pedestrians to have the same devices cars have is rather impossible, as they are unlikely to carry extra gadgets just to send safety information to the cars around. But almost all of the pedestrians circulate with one device that can communicate with other devices: a mobile phone. Some solutions in the market make use of Wi-Fi technologies; e.g., [5], where the authors use Wi-Fi both in cars and pedestrians to send beacon messages between all the agents in the road, same as in [6] and [7]. The problem of using Wi-Fi, or Wi-Fi direct, is that the communication can only be done in urban areas. Another solution is presented in [8], involving the use of cloud computing to connect cars and pedestrians. In [9] the cars are the ones that communicate the pedestrian when to be careful in road.

This document will present an architecture, an application protocol, and an algorithm running on the mobile phones, to solve the problems named above. First of all, the mobile phones analyze the pedestrian behaviour and determine when the pedestrian is creating a potentially dangerous situation, so the pedestrians will be the ones alerting the vehicles. Different sensors in the mobile phone are going to analyze three possible levels of misbehave of the pedestrians in road, taking into account proximity to road, speed, changes of the acceleration, location, and if the pedestrian is using the phone. Chapter 2 will explain the algorithm.

Once the mobile phone determines if a dangerous situation is expected, a warning message needs to be sent to nearby vehicles. Chapter 3 is the one that deals with that issue. First, an analysis of the possible road scenarios to choose the best system architecture, based on the technological requirements of the communication between cars and pedestrians. Once the architecture is defined, an analysis of security and privacy problems is done with the purpose of knowing the weaknesses of the system, to later analyze several security protocols as alternatives. All the analyzed protocols are chosen supposing the same initial criteria: the pedestrians are using their mobile phones in order

to perform the communication, and the type of messages exchange are safety ones. One or two of those security protocols will be chosen to work below the application protocol devoted to transmit the safety information.

Several technologies can be used to communicate nodes in road: Cellular (4G, LTE, 5G) or Wi-Fi, among others; and it can be a good idea to be able to use all of them independently the technology equipped in the cars. Doing so, the mobile phone can obtain information from different sources and they will be able to send and receive messages independently of the location and the capabilities of each mobile phone. Chapter 4 will compare all the different communication and secure protocols in order to choose which ones are the most suitable ones depending on the requirements analyzed in the previous chapter. After that, Chapter 5 is devoted to present the communication protocol: final version of the system architecture, emphasizing in the application protocol. After that, the security and performance requirements will meet the different solutions proposed in order to check that everything works as expected, in a theoretical way. It is supposed that vehicles will use their own communication equipment, while pedestrians will use mobile phones to perform the communication. 76% of the mobile phones in the market are Android [10], so this article emphasizes in Android environments, besides the possibility of implementing solutions because Android is open source.

Chapter 6 will present a test implementation of the different parts of the system. The implementation consists of an Android application running in several mobile phones communicating with a server via the solution proposed. There will be different qualitative and quantitative measures in order to check if the risk assumptions are done right; and an analysis of the system's efficiency in achieving the security requirements for a successful V2P communication.

Chapter 2

Receiving Road Information

The first issue to address is the detection of when a pedestrian behavior could lead to a dangerous situation on the road. For that purpose, an algorithm is designed to be used in an Android application.

This algorithm is receiving information from built-in sensors and services the Android mobile phone has, to then apply different rules determining different types of warning messages. Figure 2.1 shows the input and output of the algorithm.

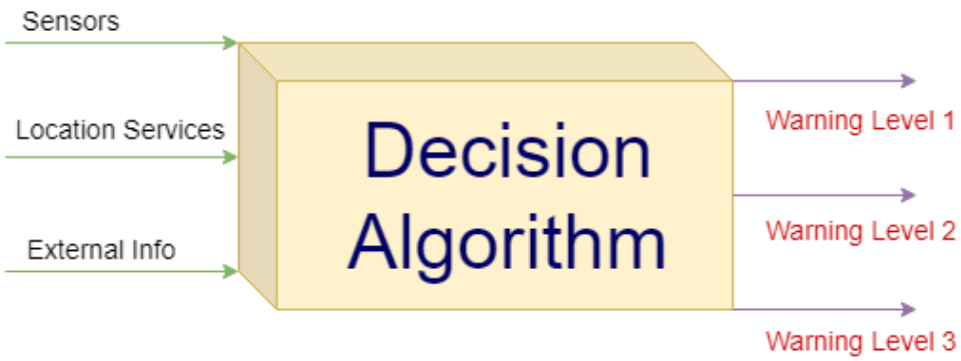

Figure 2.1: Warning-decision function

The communication part will be explained later in this document; this chapter focus on the reception of information from the mobile phone and the rules the algorithm use to determine the type of warnings. From figure 2.1, the information needed to determine the level of warning is the following:

- Sensors
  - *Acceleration*: detect dangerous movements from the pedestrian.
  - *Light*: detect if it's day or night.
  - *Proximity*: to the body, to know if the pedestrian is using the phone or not.
- Location Services
  - *Latitude and longitude*: to be able to know the area the user is in order to send the information to the proper RSU.
  - *Speed*: of the pedestrian or bike.
- External information. Messages from devices in the roadsides in order to tell the phone that the pedestrian is reaching the road.

2.1 Sensors

Most of the Android devices have built-in sensors that can measure the motion, the orientation and the environmental conditions. The precision and the accuracy of them depends on the device itself, but in any case it will be enough to obtain the necessary data we need for the application. The Android platform supports three categories of sensors [11]:

- **Motion Sensors**: measure acceleration and rotational forces along axes. From those the application is going to use the accelerometer.
- **Environmental sensors**: measure various environmental parameters, from which we are interested in the illumination.
- **Position sensors**: measure the physical position of a device. The application is going to use the proximity sensor from this group.

The sensor framework is part of the *android.hardware* package and includes the following classes and interfaces:

- *SensorManager*: to create an instance of the sensor service. The class allows the user to access and list sensors, register and unregister sensor and event listeners, and acquiring orientation information.

- *Sensor*: the class is used to create an instance of a specific sensor. The methods provided depend on the sensor's capabilities. The type of sensors the application uses are the following:
  - TYPE_ACCELEROMETER (hardware)
  - TYPE_LIGHT (hardware)
  - TYPE_PROXIMITY (hardware)
- *SensorEvent*: class created when a specific event of a sensor happens, and provides information about this event. The information provided can be the raw sensor data, the type of sensor that generated the event, the accuracy, or the timestamp.
- *SensorEventListener*: that interface creates two callback methods that receive notifications (sensor events) when sensor values change or when sensor accuracy changes. We use only the first option: *onSensorChanged()*.

The application obtains data from the sensors as shown in figure 2.2:

1. First, an instance of the sensor service is created (SensorManager). From that SensorManager, three sensor instances are created (Sensor) of the three type of sensors we want (accelerometer, light and proximity).
2. The Android Activity implements the SensorEventListener interface, so what we have to do is register one listener for each sensor we have.
3. Any time the listener receives an event (SensorEvent), the application will check which type of sensor is the one generating the event, and depending on that, it will obtain the raw data accordingly.

Now, let's proceed to explain the different sensors and how the raw data is interpreted in order to obtain the desired information.

2.1.1 Accelerometer

Let's review the axes in a mobile phone and how Android treat that data. The sensor API is relative only to the natural orientation of the screen, so the axes are not swapped when the device's screen orientation changes.

In figure 2.3 we see that axes X and Z are the ones we are more interested in, due they are the ones that determine the acceleration in the usual movement

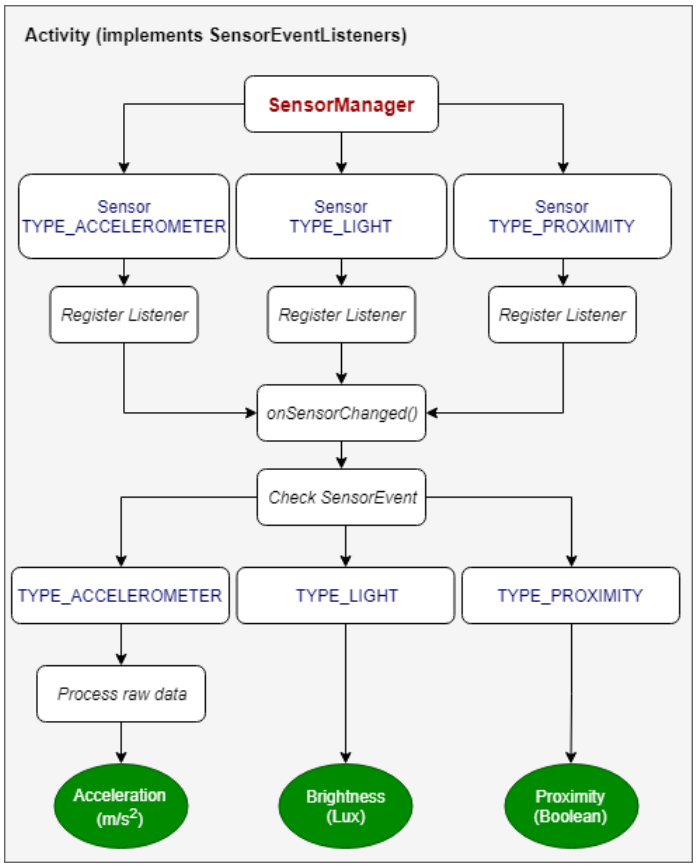

Figure 2.2: Sensors information reception diagram

of the pedestrian. Taking that into account, we can remove axis Y from the readings, but later on it will be explained how the data is treated and why the application is using all the axes.

The accelerometer sensor in Android [13] is a non-wake-up sensor that reports the acceleration of the device along the 3 sensor axes, with the gravitational forces included. All values are in SI units (m/s^2). The readings are calibrated using:

- Temperature compensation
- Online bias calibration.
- Online scale calibration.

The bias and scale calibration must only be updated while the sensor is deactivated, so as to avoid causing jumps in values during streaming. The ac-

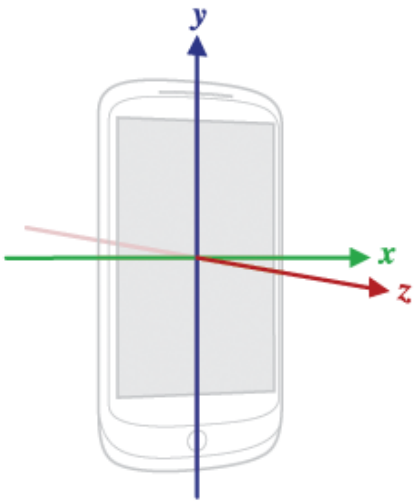

Figure 2.3: Axes in Android Devices [12]

celerometer also reports how accurate it expects its readings to be though, so that information can also be used when testing the data taken from it.

As said before, the accelerometer data can be measured both with gravity or non-gravity; depending on the sensor chosen. In the application, the accelerometer used is the one that measures the raw data (with gravity) due the higher precision; afterwards the gravity is removed in order to obtain the useful data. The acceleration applied to a device is determined by measuring the forces that are applied to the sensor itself using the following relationship:

$$A_D = -(\frac{1}{mass}) \sum F_s \tag{2.1}$$

Where A_D is the acceleration and F_s is the force applied to the sensor (we need to sum all the forces that are applied to the sensor). As we said before, the accelerometer is influenced as well for the force of gravity, so the previous equation changes like that:

$$A_D = -g - (\frac{1}{mass}) \sum F_s \tag{2.2}$$

The device will give us a magnitude of 9.81 m/s^2 (approx.) when the device is not moving on a table, and 0 m/s^2 when the device is in free fall. Therefore, the force of gravity needs to be removed. In general, the accelerometer is a good sensor to use when monitoring the acceleration. Almost every

Android device has one accelerometer, and it uses about 10 times less power than the other motion sensors. The drawbacks of the accelerometer is the use of the filter to eliminate forces, and also the possible noises.

From that sensor can be calculated as well the speed of the device, but due the small errors, after some time the accumulation of those errors will end up in a wrong lecture; so the application will use the Location Services in order to take the speed.

Calculation of acceleration

If we remove the effect of the gravity from the lectures taken from the accelerometer, the 3-axis are important to determine the acceleration, due what we want to measure are big changes in the normal movement of the pedestrian or the bike. So, the final acceleration is going to be the magnitude of the acceleration, calculated the following way:

$$A_{cc} = (x^2 + y^2 + z^2) - g \tag{2.3}$$

2.1.2 Proximity sensor

The proximity sensor lets us determine how far away an object is from a device. It is usually used to determine how far away a person's head is from the face of a device (most used to turn the screen off when someone is making a call). In most of the devices, the measurement is reported in centimeters, although some proximity sensors only support a binary "near" or "far" measurement [13]. The application is going to take into account only the second option.

The sensor is located in the face of the device, and it consists of a simple IR transmitter-receiver pair. The transmitter is a LED and emits infrared light. When placing an object close to the transmitter, the light will bounce back and the light is picked up by the receiver. In that way, it determines if something is close or near the device.

2.1.3 Light Sensor

The light sensor is a non-wake-up sensor, which reports the current illumination in SI lux units [14]. Environment sensors are hardware-based and they are available only if the device manufacturer has built them into a device. Due

Figure 2.4: Smartphone proximity sensor

the light sensor is needed to control brightness, it is available in all the devices. The sensor returns a single value for each data event, in that case lux units. Also, unlike motion and position sensors, they don't need any high-pass or low-pass filtering in order to process them.

2.2 Location Services

Pedestrians, as cars, participate in the V2P communication within their range. The Android application needs to locate the user, so the server can determine which RSU has to be reached in case a warning is needed to be sent. In addition, as we said before, the speed of the pedestrian is difficult to determine using the accelerometer, due to the small errors in the readings that would result in a wrong speed lecture, so the Location Services can also give us that information.

In Android [15], we can use both GPS and Android's Network Location Provider to acquire the user location. The challenge here is to decide which one is better to use, in other terms, the strategy to follow when getting the user's location. First, let's check the characteristics of each solution:

- GPS
  - Most accurate.
  - Only works outdoors.
  - Quickly consumes battery power.
  - Does not return the location that quickly.

- Android's Network Location Provider
  - Use cell towers and Wi-Fi signals.
  - Provides location information both outdoors and indoors.
  - Responds faster.
  - Uses less battery power.

From the information taken above, GPS seems to be the perfect solution to determine the location of the pedestrian and the speed; due the accuracy is the best one and we don't need indoors readings (pedestrians are not dangerous if they are inside the buildings). The only two issues that need to be solved are the battery power and the pedestrian location during the first seconds of the performance.

The battery issues is really difficult to solve, because we need to know the location of the user and the speed in any moment to know if a dangerous situation is taking place. The second issue can be solved easily, using two strategies:

- Reading the info from the Network Location Provider during the first 2 minutes, and then change to the GPS.
- Taking the last location information known, supposing the user has enabled the location always.

The way to proceed with the reception of location information can be seen in Figure 2.5, and works as follows:

1. First, we create an instance of the location services: *LocationManager*. In addition, we create an instance of the location listener: *LocationListener*.
2. The *LocationListener* instance implements several methods, but the application is going to use only one: *onLocationChanged()*, which is going to obtain a class *Location*.
3. Register the *LocationListener* into the *LocationManager* asking for updates. Those updates can be asked with no delay or after some seconds or meters walked (depending on the selection of these variables, the application can also save some battery).

4. When the location is updated and a change of location is triggered, the Location class is treated to obtain the latitude, longitude, and speed (km/h).

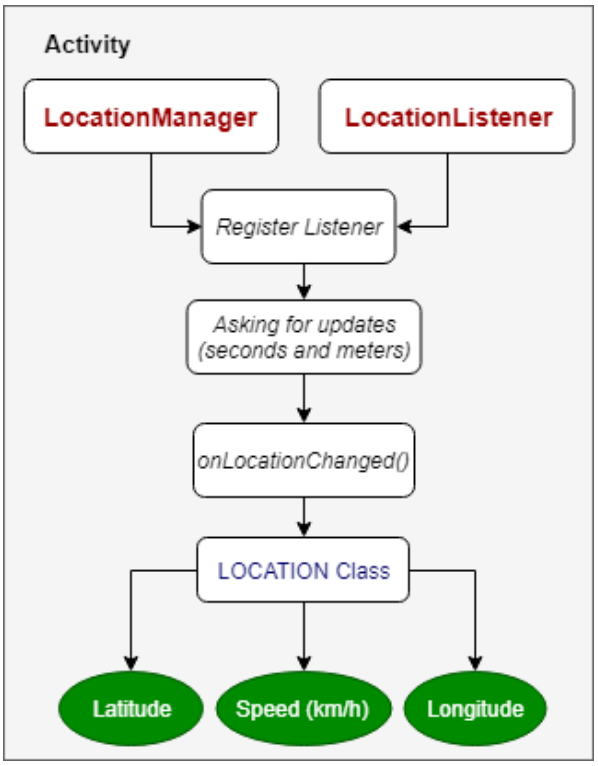

Figure 2.5: Location information reception diagram

2.3 Proximity to Road

In the previous section, the Location Services were presented and GPS was the option chosen in order to locate the pedestrian. Using either GPS or the Android's Network Location Provider can give us a good result when trying to locate an user in a corresponding area.

However, when trying to know when an user is close to the road is challenging. The solutions presented in chapter 1 that address the safety detection of the pedestrians with the mobile phones use the cameras as a way to detect when the pedestrian is close to the road (either to cross or just passing by). As

said before, these solutions are out of the scope of this project due the necessity to have the mobile phone always out of the pockets in the hands, as well as due to privacy reason.

Using the location given by Android may present several errors. In the Android documentation [16], they defined accuracy as the 68% confidence based on the *horizontal accuracy*. That means, inside the circle with radius the horizontal accuracy, there is a 68% of getting the right location. Although the horizontal accuracy can be set - by the method *setAccuracy()* -, that does not mean that the value set is the one used, it depends on the location. Some work was done in order to determine how precise can be the GPS information given by Android in [17]. They used the GPS information and a maps database in order to check if the information can be trusted in order to determine the pedestrian distance to the road. They found that in rural areas their experiments showed promising results, on the contrary than in urban and suburban areas due to the large errors in positioning and delays in detection.

For that reason, we are going to take another approach and make one assumption: that in the sidewalks there are small transmitters that will notify the mobile phone when it is close to them. The mobile phone will just detect these beacons when it is close enough to the road, notifying the decision algorithm that the user is possibly trying to cross the road, or at least in a critical location. In order to introduce that communication, several options of short-range wireless communications can be used: Bluetooth, ZigBee or Infrared. The problem with ZigBee is the need to introduce external hardware to the mobile phones in order to allow the communication, and Infrared is a technology that not all of the actual mobile phones have implemented. That leaves us with Bluetooth as technology to use as the communication between road-transmitters and mobile phones.

2.3.1 Bluetooth LE

The technology used to receive the beacons from the road transmitters is Bluetooth Low Energy [18] - Bluetooth LE or BLE -. It was designed in 2011 by the Bluetooth Special Interest Group (Bluetooth SIG) for applications in healthcare, fitness, beacons, security, and home entertainment.

The key difference with Bluetooth is the low power consumption, a very positive aspect when dealing with short-range and M2M (machine-to-machine)

communications. Devices running under BLE can have a small battery and still work for 4 or 5 days without problems. BLE operates in the 2.4 GHz ISM band, as Bluetooth, although BLE remains in sleep mode constantly except for when a connection is initiated. With a data rate of 1 Mb/s, the connection in BLE is only a few miliseconds (in Bluetoogh can take around 100 miliseconds). That is why that technology is perfect to use in our schema: the transmitter devices will send every fixed time a beacon message, that Android mobile phones can obtain using the Bluetooth receiver. BLE is used due we don't need to exchange large amounts of data (just a beacon message) so therefore the battery power can be saved, and the devices don't need to pair before the exchange of data.

In order to develop the reception of the beacons in the mobile phones, Android introduces a built-in platform support for BLE in Android 4.3 (API level 18) providing the applications to discover devices, query for services and receive/transmit information. Different concepts should be defined in order to understand how BLE works:

- *Generic Attribute Profile (GATT)*: the GATT is the general specification for sending and receiving the data, also known as "attributes", over the BLE link. All the BLE application profiles are based on GATT, and Bluetooth SIG defines several profiles for BLE devices. A profile is the specification of how a particular device works depending on the application, and a device can have more than one profile. One example can be a device that contain a temperature monitor and a battery level detector.

- *Attribute Protocol (ATT)*: the GATT is built on top of the ATT. Each attribute is uniquely identified by a Universally Unique Identifier (UUID), which it is standardized in a 128-bit format string ID. The attributes transported by the ATT are formatted as characteristics and services.

- *Characteristic*: it contains a single value from 0 to "n" descriptors that describe the characteristics value (the type).

- *Descriptor*: they are defined attributes that describe a characteristic value. For example it can specify a human-readable description, an acceptable range for a characteristic's value, or a unit of measure.

- *Service*: it is a collection of characteristics. For example a service called "Temperature Monitor" can include characteristics like the "temperature measurement".

Two can be the topologies than can be followed when Android devices and BLE devices interacts between each other:

- *Central/Peripherical*: applied to the BLE connection. The device that has the central role scans looking for advertisement, while the device with the peripherical role is the one making the advertisement. The communication can be done only between a central and a peripherical node, not between two central or two peripherical nodes.
- *GATT server/GATT client*: applied to how the two devices talk to each other once the connection is established.

In our schema we have an Android phone receiving the beacons of proximity to the road, and a BLE device sending those beacons. The phone supports the central role, and the BLE device has the peripherical role. Once the phone and the device have established the connection, they can start sending GATT metadata. The BLE devices are the ones sending the beacons, so they are going to act as server, while the mobile phones will be the clients.

2.3.2 Beacon Reception

We suppose the BLE devices in the roadside are sending permanently (or every fixed time) their unique identifier (beacon information) to notify the nearby mobile phones that they are close to the road. When the Android application receives that message, it is going to notify the decision algorithm about the proximity to the road, as it will be seen later in this chapter. The Android SDK has no built-in support for beacons, only the reception of BLE information, although there is a library from Radius Networks [19] that can be used.

The library allows the Android devices to use beacons by requesting notifications when one or more devices appear or disappear. The application can also request to get a ranging update from one or more beacon devices, with a frequency of approximately 1 Hz. The library also supports the capability of the Android phones to send beacons. By default, the beacons detected are the ones meeting the open AltBeacon standard [20], so we assume all the beacon devices needed in our system are following AltBeacon standard. The library also allows to determine the distance in meters with the beacon.

Figure 2.6 shows how the communication with the BLE devices works. First, the *BeaconManager* is created and binded to the activity (which has

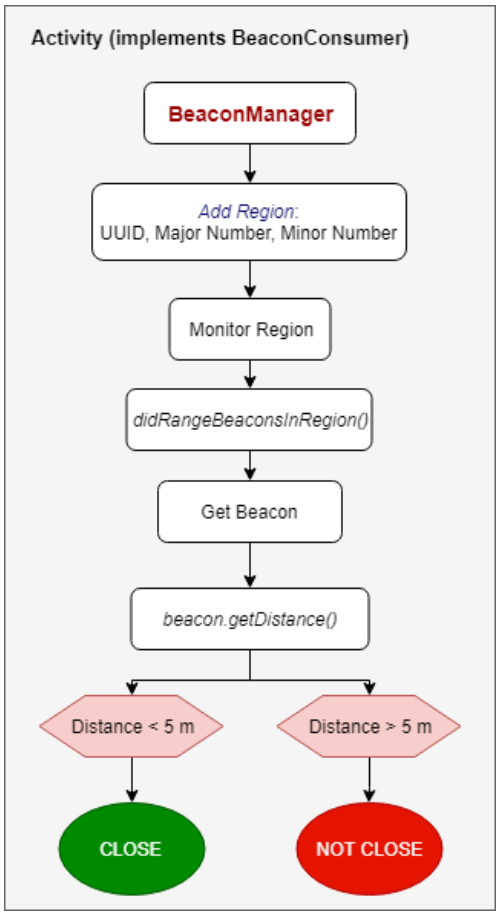

Figure 2.6: Beacon reception diagram

to be implemented *BeaconConsumer*). At this point, the Bluetooth receiver is able to receive any communication, so the procedure to detect the beacon devices is the following one:

- The Region is defined. "Region" in this context means the number of services or attributes that the mobile phone is going to monitor. In our case, the *Region* is formed by only 1 attribute. The identification of an specific attribute is done by an ID formed of the following information:
  - UUID: 128-bit unique identifier from the organization or company of the device.
  - Major and minor number: 16-bit unsigned number, that can be provided by the store or by the owner of the devices. With those 2 numbers, an specific beacon service can be defined to be differentiated from the other ones.

- Start monitor the *Region*. As soon as any beacon device with the specific ID is within the range, the method *didRangeBeaconsInRegion()* triggers.
- That method is returning a list of *Beacon* classes, corresponding to the beacon devices in the range. With the method *getDistance()* the distance of the mobile phone and the beacon device is given.
- If the mobile phone is obtaining data from a device, and the distance of this device is lower than 5 meters, we can conclude that the user is approaching the road.

2.4 Decision Algorithm

2.4.1 Input Information

In this section, the decision algorithm is explained. From the information given by the sensors, the location information and the external sources, some rules are applied in order to determine the warning level the application is going to send the server. Table 2.1 summarize the different data obtained and used by the algorithm.

Source	**Information**	**Magnitude**
Sensors	Acceleration	m/s^2
	Light	Lux
	Proximity	Boolean - FAR/CLOSE
Location Services	Location	Latitude & Longitude
	Speed	km/h
External Sources	Proximity to road	Boolean - CLOSE/NOT CLOSE

Table 2.1: Input information of the Decision Algorithm

2.4.2 Output of the algorithm

Having all the possible inputs of the algorithm in table 2.1, we need to define some rules in order to identify which type of warning the application is going to send to the server. Depending on the mode of action (walking, running or biking) and the information taken from the pedestrian (location, acceleration, speed, use of mobile phone, and relative position to the road), the rules are going to analyze if any dangerous situation is being created by the pedestrian.

Mode	Parameter	Value	Danger
Walking	Min. Speed (km/h)	4	NO
	Max. Speed (km/h)	10	>10 km/h
	Change Acc. (m/s2)	2	>2 m/s2
Running	Min. Speed (km/h)	8	NO
	Max. Speed (km/h)	20	>20 km/h
	Change Acc. (m/s2)	1	>1 m/s2
External Sources	Min. Speed (km/h)	7	<7 km/h
	Max. Speed (km/h)	30	>30 km/h
	Change Acc. (m/s2)	3	>3 m/s2

Table 2.2: Parameters of dangerous road situations

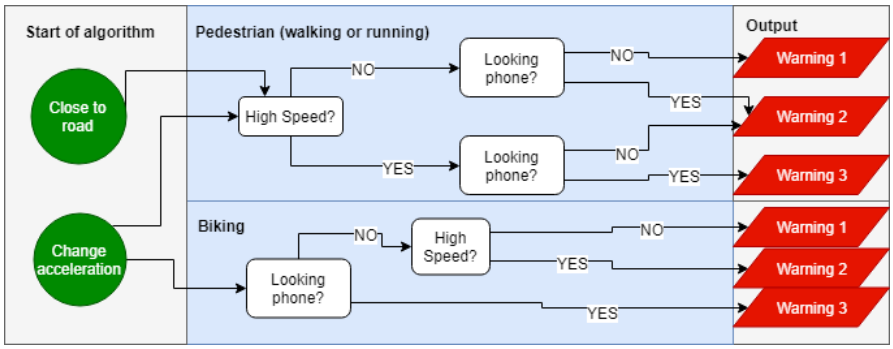

Figure 2.7: State diagram of the decision algorithm

- Warning Level 3 situations:
  - The pedestrian is doing dangerous movements close the road while looking at the phone and going faster than expected.
  - The cyclist is looking at the phone while biking.
- Warning Level 2 situations:
  - The pedestrian is doing dangerous movements close the road while looking at the phone or going faster than expected.
  - The cyclist is going faster than expected.
- Warning Level 1 situations:
  - The pedestrian is close to the road and looking at the phone.
  - The cyclist is changing speed too fast.

These rules are easy to read by a human, but the application need some data comparison in order to determine what is “dangerous movement” or “faster”, for example. Table 2.2 defines some parameters in order to translate the rules above into "boolean" terms; also figure 2.7 shows the state diagram of the decision algorithm for pedestrians and bikers.

The last thing to take into account from the data obtained is the brightness. In that case, if the phone detects that the pedestrian is using the application during the night, the warning level is incremented by 1 (having only level 2 and level 3 possibilities). From Lux values, we can determine if in an outside place is day or night [21]: below from 100 Lux, we can say it is night.

Chapter 3

System Architecture

We consider a V2P situation where the goal, eventually, is to eliminate the crash scenarios (and their fatality's probability) between cars and pedestrians, by communicating in **pre-crash scenarios**. Most of the V2V or V2P solutions try to avoid crash scenarios, where cars or pedestrians try to avoid the collision by processing beacons sent by all the agents in the communication.

The idea is to inform the vehicles that a possibly dangerous situation may happen so they can reduce the speed or the drivers can pay more attention. That will decrease the chance to have a fatal collision, or a collision at all. Alternatively, the safety beacons that the vehicles are sending (in a normal V2V safety situation) can be received and processed by pedestrian, helping to the overall performance of the solution. Four main node types are the ones participating in the communication [22] [23]:

- **Vehicle device**: vehicles communicating between each other using V2V communication, and with the other networks using the infrastructure as access point - V2I communication.
- **Road Users**: those are the vulnerable users in the scenario: pedestrians or bikers. They use their mobile phones to communicate with the Internet, and in our solution with the cars nearby.
- **Infrastructure**: road infrastructure, such us traffic lights or lane markings, that can communicate with the vehicles and pedestrians providing road information, or used as access points to the Internet - in the case of V2I communication to provide services -.

- **Processing Unit**: although this node can be implemented in the vehicle devices and road users phones, when talking about indirect communication it reduces the amount of processing of the other nodes. The idea of this component is to send/receive the safe messages, and determine when there is a dangerous situation in road.

Three are the different modes of communication in this scenario, depending on where the processing unit is developed: direct, indirect, or hybrid communication. The **direct communication** involves a data exchanged between the vehicle devices and the road users without intermediate entities. Due there is no third party, this mode can be seen as the best one for safety applications in terms of latency. However, now we are dealing with different type of nodes both in terms of technology used and road behavior, so several problems arise that makes the direct communication not the suitable one.

First of all, *the devices must use the same technology*. V2V protocols can use direct communication due the vehicles are equipped with devices that use same communication protocols, but when pedestrians enter in the system either they have to use the same devices, or introduce backwards compatibility in the V2V protocols. Use the same device is very difficult to achieve. As an alternative, in [24] the authors proposed a solution using DSRC in the mobile phones to communicate with the cars, but in order to make that happen they have to change the firmware of the Wi-Fi chip in order to adapt the 5GHz band to the range between 5.85GHz - 5.925GHz that DSRC uses. And the use of extra gadgets equipped with the same technology as the cars is really unnecessary. The backwards compatibility solution is just inefficient, all the work done in securing V2V protocols can be tore down when trying to make those changes.

The second problem of direct communication is the *necessary computing power*. Already both vehicle devices and mobile phones should process the road information by using sensors or by obtaining information from the infrastructure nodes. Adding also the processing of the safety messages and the detection of dangerous situations will increase the power needed in the devices, producing delay in the processing and therefore in the delay of the responses. In addition, the big amount of pedestrians in the road may increase the processing times in the vehicles, having to check if their messages pose a real danger.

The *range of the communication* is the third problem in the direct communication. One example is the use of Bluetooth, where the speed and the distance between the devices are against a proper direct communication. One solution can be the multi-hop communication, but the technology incompatibility makes that hard to implement. The last problem with the direct communication is not related to the nature of the direct link, but with the *different road behaviour of the nodes*. We are dealing with pedestrians and vehicles, two different types of nodes which movement, speed, and behavior in road is very different. Most likely, the solution will be used in urban and suburban areas, where the amount of pedestrians is way higher than the vehicles. Besides, the pedestrians are using their mobile phones which can produce several wrong safety messages in some situations. Let's imagine 10 pedestrians trying to cross at the same time; in this case 10 safety messages will be sent, although with 1 message to the nearby cars would be enough.

The other mode of communication will help to solve most of the problems commented above: **indirect communication**. In that mode, the processing unit is located between the vehicle devices and the pedestrians, so all the processing is done by that third node, removing the problem of the range. Also the technology incompatibility is solved: vehicles can communicate between them using V2V and V2I to the processing unit, and Wi-Fi or cellular communication between Android mobile phones and the processing unit. The processing unit can process some messages of the same type and in the same location as one, reducing the amount of warnings that the cars receive. A variation of this mode can solve the NLOS scenarios, by re-broadcasting the safety messages between devices in the same network. There is only one problem that this mode presents, and it is a crucial one: the delay of the communication will be higher, so the communication protocol used should try to reduce it.

Nevertheless, the perfect architecture in the scenario is a **hybrid** one: a processing unit of the data exchanged between vehicles and pedestrians that will process most of the safety messages sent by the pedestrians, but some of those messages (the most critical ones) are sent directly. Re-broadcasting between nodes in the same network will help to solve NLOS situations, and to reduce the number of messages send by the nodes. Figure 3.1 shows the hybrid V2P schema: mobile phones communicate both with a processing unit and the VANET, and there will be some infrastructure communicating road information to either vehicles or mobile phones (direct transmission). Mobile phones communicate directly with vehicles only in special occasions, and the

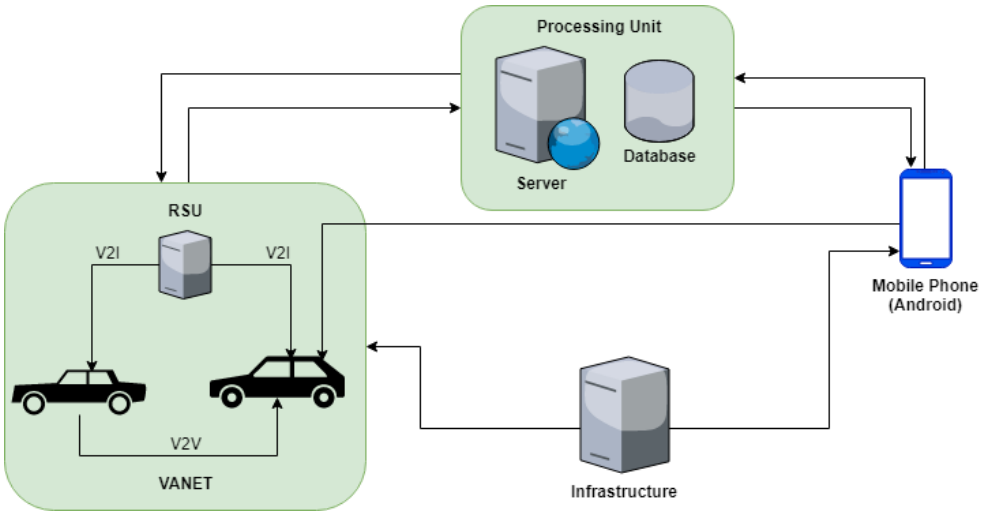

Figure 3.1: Hybrid V2P schema

processing unit works as filter in the sense of removing wrong safety messages and reducing the amount of processing time by the vehicles.

An important concern, before dealing with the security and privacy, is the delay between the broadcast of messages by the pedestrians and the reception of a safety message by the vehicles. The idea is to act in the pre-crash scenarios: try to identify when an user is creating a dangerous situation, and try that the other users (specially vehicles) are aware in order to avoid critical accidents. In direct scenarios, that delay should be as small as possible (same as in V2V situations) although it can be a little bit higher: less than 50 ms (near-real time). In indirect scenarios, the main delay has to come from the processing unit and its action time: we suppose that in a pre-crash scenario the minimum distance between the car and the pedestrian is 15 meters and the mean speed of vehicles in urban scenarios is 40 km/h; so between 1 and 2 seconds should be a good value for the delay.

3.1 Security & privacy requirements [25] [26]

1. **Confidentiality**: the content of a message is kept secret from those nodes that are not authorized to access it. Attackers that read the packets exchanged via the wireless network should not have any information about the user, or the sensitive information exchanged between parties.

2. **Entity Authentication**: the receiver is not only ensured that the sender

generated a message but in addition has evidence of the liveness of the sender. The system should corroborate the identity and legitimacy of the nodes. If the service were open to any user, without checking the credibility of its actions, that user could attack the system by sending false safety messages, resulting in creating a chaos in the traffic, or even accidents.

3. **Message Authentication and Integrity**: messages must be protected from any alteration and the receiver of a message must corroborate the sender of the message. Integrity, however, does not necessarily imply identification of the sender of the message.

4. **Access Control**: access to specific services provided by the infrastructure nodes, or other nodes, is determined locally by policies. Although the access to the messages and the network should be open to all nodes, some control is needed in order to avoid bad practices. As part of access control, authorization establishes what each node is allowed to do in the network, e.g., which types of messages it can insert in the network, or more generally the protocols it is allowed to execute.

5. **Non-repudiation**: the sender of a message cannot deny having sent a message. Both the mobile phones and the server should be capable of proving that they are the ones sending the data. An attacker can act as one of the pedestrians sending wrong data about their location or the road situation, or acting as a fake server and disabling the service.

6. **Availability**: protocols and services should remain operational even in the presence of faults, malicious or benign. This implies not only secure but also fault-tolerant designs, resilience to resource depletion attacks, as well as self-stable protocols, which resume their normal operation after the 'removal' of the faulty participants. An attacker can try to delay the service by sending DoS or DDoS attacks to any of the nodes, so the processing times increase and the overall delay is above the limit for a near-real time exchange of data. In an indirect communication, the server is a single point of failure that needs to be protected of attacks, and countermeasures should be implemented in case that server is not available.

7. **Liability Identification**: users of vehicles and pedestrians are liable for their deliberate or accidental actions that disrupt the operation of

other nodes, or the transportation system. The network should provide information that identifies or assists the attribution of liability.

8. **Privacy**: this is one of the most critical requirements that the solution must provide. We require anonymity for the actions (e.g., messages, transactions) of the vehicular network entities, with respect to a set of observers. At minimum, any of the observers should not be able to learn if a node performed or will perform in the future a specific action, assuming that the node performs the action. Such a definition does not, however, guarantee that it is impossible for the observer to infer, with relatively high probability, the identity of the node that performs the action in question. In our particular case, the location or the speed of the pedestrian are key data that not even registered users in the system should know. Variables related to the behaviour of the user in the road can be send private by using an application that does not send such data, but key variables that can make the processing unit to know the behaviour. In such case, the location is the only, but very important, sensitive information sent by the pedestrian. The service should be capable to not store that location in a way to be able to track the user, or to know which user is the one sending a message from a specific location.

Chapter 4

Protocol Alternatives

Let's discuss the different communication protocols that may work in the implementation of the data exchange between the pedestrian and the processing unit or the pedestrian and the vehicle, under the requirements discussed before.

4.1 Direct communication protocols

Within this group, all the communication protocols are the ones used in V2V networks and M2M - machine to machine - networks. Also Bluetooth is analyzed in this group because the communication is direct, it is assumed that vehicles have the capability to use it, and Android has Bluetooth libraries to implement it in any application.

4.1.1 DSRC/WAVE

DSRC [27] and its wireless component, WAVE, are the group of standards that provides the architecture for VANETs. DSRC/WAVE is used by vehicles with GPS and equipped with OBUs (On-Board units) to propagate safety warnings in V2V communications. It operates in the 5.9 GHz band and has 75 MHz of BW allocated for vehicle communication, based on LOS with speeds up to 140 km/h and 1 km of range. From all the standards in WAVE (still in development), the one that describes the security measures is the IEEE P1609.2 standard [28].

- **Confidentiality**: depends on the type of message: broadcast messages or transaction messages. Due the pedestrians are going to send safety messages to cars, those are broadcast ones. In the standard, those messages are not encrypted but signed with the sender's certificate. Due the

location is not needed to be sent in direct communications, no sensitive information is shared and the confidentiality of the information is guaranteed without encryption.

- **Authentication**: senders sign the broadcast messages with theirs digital certificates, using ECDSA (Elliptic Curve Digital Signature Algorithm) as signature algorithm. The standards recommends in RSUs the enrollment of certificates including a manual component, but for OBUs (and in our case mobile phones) there is not a procedure to enroll certificates. When signing a message, the mobile phones may include a certificate chain; vehicles must verify the message by recognizing the root certificate used by the sender's certificate. Certificate Revocation Lists (CRLs) are also used. From the standard, authentication is provided by the use of VPKI solutions that can handle digital certificates to the nodes.

- **Integrity**: provided by the use of digital certificates, so it is provided with the use of VPKI solutions.

- **Non-Repudiation**: the standard does not deal with the liability identification. On the other hand, due the messages exchanges are safety ones (broadcast) along with the use of CRLs, meeting the requirement of non-repudiation is not an important matter.

- **Availability**: threats to the availability of the system are the introduction of malware, DDoS attacks, or spamming messages [29]. The solutions already applied in traditional wireless networks need to be tested in order to check if they can work in these scenarios.

- **Privacy**: although in the safety messages, the location is not shared, location tracking is an issue that needs to be solved. The nodes must be identified in order to identify and stop attackers, but on the other hand the network should allow to send broadcast messages anonymously. There is no mechanism in WAVE allowing anonymous broadcast messages, so the solution of this aspect should rely in external solutions.

Let's discuss briefly the technical aspects in order to use DSRC/WAVE in mobile phones. Although in terms of delay the solution is the best one, mobile phones should change their hardware or firmware in order to implement the solution. In addition Android does not include DSRC libraries, so the protocol should be implemented in the code of the safety application from scratch.

4.1.2 M2M protocols

Machine-to-machine protocols are mostly used in IoT and IIoT systems, and they can be useful when talking about communication between mobile phones in roads, or between cars and mobile phones. Two are the main M2M communication application protocols: MQTT and CoAP.

- *Message Queuing Telemetry Transport, or MQTT*. It is based in a publish-subscribe protocol that enables one-to-many communication mediated by "brokers". The clients publish their messages to a broker and/or subscriber to a broker to receive certain messages.
- *Constrained Application Protocol, or CoAP*. It is client-server based, where the client nodes can command other nodes by sending packets that will be interpreted by the server node.

In IIoT, MQTT is preferred over CoAP for mission-critical communications due it ensures the delivery of the messages; although CoAP is preferred in data transmission of low-power nodes. Both protocols are quite difficult to implement in our schema, due the mobility of the vehicles and the pedestrians, and both architectures needs a node acting as a server. Using the typical architecture of a VANET the RSU can be that node, although some assumptions have to be made. In case of MQTT, vehicles should ask periodically for the messages post by the pedestrians, so that protocol is not very useful for the purpose of safety messages. On the other hand, in CoAP, pedestrians can send the messages directly to the RSU or the vehicles - acting as servers - so let's analyze a little bit more CoAP to check if it can be a good solution to send direct messages from pedestrians to vehicles.

RFC7252 [30] is the document where CoAP is presented. As said before, CoAP is a client-server communication protocol based on the RESTful structure. like HTTP. Its architecture is divided into two layers: the message layer and the request/response layer. The first layer is the one in charge of the UDP exchange between the two points, and the second one is in charge of the methods sent. UDP is the transport protocol used, and the security aspects are solved by DTLS - Datagram Transport Layer Security - and sometimes by IPSec. DTLS AES/CCM mode provides confidentiality, integrity, authentication, and non-repudiation; although there are some issues [31] that may not ensure all the security requirements listed before:

- **Headers**: DTLS header adds 13 bytes per datagram, so they can be longer than the network MTU; that without counting the handshake anytime a datagram is sent - high delays -. One solution implemented in [32] compresses the header to 3-5 byes by using 6LoWPAN, although the authors did not ensure that the compression would not affect the security. Furthermore, using that compression removes the possibility to use CoAP proxies.

- **Key Management**: from all the security modes that CoAP defined, two of them works with PKI: *RawPublicKey* and *Certificates*; both supported by ECC - Elliptic Curve Cryptography - as public key cryptography. The big problem that is not solved is the key management, a problem that is common also in DSRC.

In summary, the main issues with CoAP is the heavy cost of computation due the handshakes, the possibility of having message fragmentation, and the key management. The message fragmentation can be solved by the use of 6LoWPAN to compress messages, although the security is not assured with that solution.

In mobile phones, *Californium* is a CoAP framework that uses JAVA; although the implementation is based in a cloud-based client and server architecture. Other implementations used in low-power devices communication use other languages - mainly C -. That means, in Android can be implemented, although with some limitations. Besides, it needs to be implemented from scratch in VANETs.

4.1.3 Bluetooth

Bluetooth is a short range data transmission technology, used for data transmission from one device to another, or one to more devices. The fact that Bluetooth devices can support multiple data rates and that the technology is included in smart-phones, makes it a nice option to use to communicate vehicles and pedestrians directly. The only problem is that the common Bluetooth structure connects devices in ad-hoc networks after the establishment of a link key, adding a long delay in the process of connection. For that reason let's talk about Bluetooth 4.0 and in advance, that allows the communication between devices using Low-Power or Low-Energy Bluetooth, BLE [33].

The main difference of BLE from other versions of Bluetooth is the low power consumption. BLE operates in the 2.4 GHz ISM band, as Bluetooth, although BLE remains in sleep mode constantly except for when a connection is initiated. With a data rate of 1 Mbps, the connection in BLE is only a few miliseconds (in Bluetoogh can take around 100 miliseconds). About the security and privacy, the main issues that BLE faces are in the pairing process, although there are another possible attacks that BLE can face [34]:

- **Passive eavesdropping**: third party listening to the data exchanged between nodes. BLE tries to solve that issue by using AES-CCM cryptography. Although AES is very secured, the pairing method BLE uses - how the keys are exchanged - introduces several vulnerabilities that allow attackers to decrypt the data. The confidentiality is no longer ensure.

- **MITM**: third party impersonating one of the other two nodes exchanging data, in order to make believe that it is one of the trust nodes. As before, the pairing method introduces that vulnerability. Non-repudiation is not ensured.

- **PKI**: in order to ensure authentication and integrity, a certificate needs to be used. As in other solutions, a PKI solution has to be used.

- **Identity tracking**: when an attacker can associate the address of a device with a specific user and therefore tracking that user. The way BLE solves this is by periodically changing the device address, using *Random Private Resolvable* or *Random Private Non-Resolvable* type of addresses. The first option uses the IRK (Identity Resolving Key) method, which translates its device address into a random one. A second device with IRK as well is able to translate back the random address into the sender's one. That random address is generated periodically. In the second option, the device address is just a random number.

- **DoS**: Bluetooth is vulnerable to DoS attacks by trying to create fake pairing processes and disabling the capability of that device to pair other devices. Also another attacker can start sending fake data to the vehicles, creating dangerous situations in road.

The last part of the analysis is the possibility of implementation of Bluetooth and BLE in the vehicles and pedestrians. Android introduces a built-in platform support for BLE in Android 4.3 providing the delivery of beacons

between devices, so some changes are needed to exchange messages between the nodes.

4.2 Indirect communication protocols

Once the direct communication protocols have been analyzed, it is the turn of the protocols of the indirect communication protocols. The indirect communication will try to filter some of the data sent by the pedestrians, and warn the vehicles only when a significant number of pedestrians are generating a threat in a certain area. The protocols to analyze are security protocols, one acting in the network layer (IPSec) and other three introducing security solutions in the transport protocol. Although the favourite transport protocol in our schema is UDP, due its capacity to send datagrams with a lower delay than TCP; the security protocols analyzed are both solutions for TCP and UDP.

4.2.1 IPSec

IPSec is a set of secure network protocols, used mainly in VPNs. About its architecture, IPSec has two different operation protocols, with some differences in the security that they provide:

- *Authentication Headers - AH [35]*: it provides connectionless data integrity, using a hash function and a secret shared key; and authentication of the data origin. AH does not provide confidentiality, being useful when the only concern is the modification of the data and to reduce processing times.
- *Encapsulating Security Payloads - ESP [36]*: it provides data integrity, confidentiality and authentication of the origin.

In order to set the parameters and algorithms to use by the nodes to operate either with AH or ESP , IPSec uses *Security Associations*, SAs [37], by setting up a session. That SAs are established using the ISAKMP (Internet Security Association and Key Management Protocol), which is implemented using manual configuration using pre-shared secrets. In order to manage those SAs each node has to implement a SA Data Base, SADB, keeping the record of which encryption and hashing algorithm the host can use, and the ports/nodes with permissions to communicate. IPSec can work as well in two different

modes: Transport and Tunnel Mode; which define if the whole packet is encrypted and authenticated, or only the payload. From the information collected, IPSec solves some of the security requirements in the V2P communication: confidentiality, integrity and authentication; although its set up needs a SADB. The problem with that is the need of pre-shared keys and the complexity of a big amount of keys and sessions opened only to send safety messages. Moreover, there are two important aspects to take into account about IPSec vulnerabilities:

- **Known attacks**: as exposed in [38], Vesselin Tzvetkov from Bochum University in Germany talks about two attacks against IPSec in its Tunnel Mode, although the replay attack countermeasure that IPSec implements can make them hard to succeed.
  - Cut-And-Past Attack.
  - Session Hijacking.
- **Underlying protocols vulnerabilities**: that is the major problem with IPSec. There are some underlying technologies that are needed in order to achieve the authentication and the encryption, for example the establishment of the SAs is defined by external protocols. Vulnerabilities on those key exchange methods affect the security of IPSec.

Let's now talk about the implementation requirements. There are several open-source libraries implementing IPSec VPNs that can be used as well in Android devices, such as *strongSwan*. The library *IpSecManager* of Android has methods for managing IPSec sessions, although not all the aspects are implemented. About the delay IPSec introduces, neglecting the time involving the SAs due it is supposed the sessions are up when the safety messages are sent, the ESP header is up to 57 bytes. The safety messages does not involve more than 20 bytes of data, so this solution may introduce high delays in an indirect communication.

4.2.2 SSL/TLS

Secure Sockets Layer - SSL [39] - and the new version of it, Transport Layer Security - TLS [40] -, are cryptographic protocols designed to provide communications security. The encryption used in TLS uses symmetric cryptography, and the unique shared keys used are exchanged during the TLS handshake (using asymmetric encryption). Doing that, TLS uses the safest encryption

method to exchange common data, and symmetric cryptography reduces the use of computing resources. The handshake is quite complex:

- The client initiates the secure connection, and the server replies with the list of cipher suites that it knows. The client checks the cipher suites it can use, and lets the server know the one both are using.
- The server provides its *digital certificate* containing the server's public cryptography key. The client checks the certificate's authenticity.
- Client and server establish a session key, with the help of the server's public key, that they will use to encrypt the communication.

For now, confidentiality and integrity are ensured with the use of the symmetric cryptography, establishing that shared key using asymmetric cryptography. Privacy is ensured by the use of session keys, that are not linked to any user. However, the authentication and non-repudiation, in addition to the good performance of the overall security protocol, go through the use of **digital certificates**. A digital certificate can be seen as an electronic "password" that allows to exchange data securely over the Internet using the PKI. From the digital certificate, a private key and a public key can be obtained. Depending if we want to encrypt data or to sign data, the use of these keys are different:

- *Encryption/Decryption*: the application that want to send a secure message uses the public key of the other application; and the receiving party decrypts the message with its private key.
- *Sign/Verify*: the sender signs the message with its private key, and the receiver verifies that the message was sent by the sender using the public key from the sender.

TLS typically uses certificate authorities in order to establish the authenticity of the certificates. Trust is usually anchored in a list of certificates distributed with user agent software, and can be modified by the relying party. Therefore, a PKI solution is needed to be able to ensure almost all the security requirements of the solution proposed using SSL/TLS.

SSL/TLS works over TCP so the exchange of messages is connection-oriented with the use of ACKs, which may introduce some delay in the exchange of safety messages from the pedestrians to the server. An application protocol that can be used over TLS is HTTPS, based on RESTful methods. Some vulnerabilities known [41] in TLS and therefore HTTPS are the following:

- **POODLE**, Padding Oracle On Dowgraded Legacy Encryption, published in October 2014. It takes advantage of the interoperability and compatibility between versions, and block padding - a vulnerability in SSL 3.0 -. An attacker can make a MITM attack, impersonating the server and making the client to use SSL 3.0. The vulnerability in that version is in the CBC mode. An attacker will be able to encrypt and decrypt blocks by modifying padding bytes and checking the server response; taking a maximum of 256 requests to decrypt a single byte. That vulnerability can be solved by having the latest version, TLS.

- **BEAST**, Browser Exploit Againts SSL/TLS, published in September 2011 and affecting SSL 3.0 and TLS 1.0. The attacker impersonates a client injecting packets into the TLS stream (MITM). The attacker will guess the Initialization Vector (IV) used with the injected message and comparing the results to the ones of the messages the attacker wants to decrytp. It can be solved by using TLS 1.1 or TLS 1.2.

- **BREACH**, Browser Reconnaissance and Efiltration via Adaptive Compression of Hypertext, is a vulnerability affecting TLS via targeting the HTTP compression. The attacer will force the victim's browser to connect to a third party website (with TLS enabled) and monitoring the traffic between that server and the victim. The prevention of the attack can be done by changes in the HTTP configuration, like disabling the compression or limiting the rate of requests.

As seen from the vulnerabilities exposed all of them can be prevented by using the latest version of TLS. Therefore, the security requirements exposed in chapter 3, except availability, can be met with the use of a PKI solution. About the implementation requirements, Android provides good and reliable libraries based on Java to work both with SSL/TLS (SSLSocket) and HTTPS (HttpsURLConnection). About the delay, let's neglect the overhead introduced by the session establishment due it can be done before the exchange of safety data, so the interest relay in the overhead for the encrypted application data. The data is carried in TLS Records with 5 bytes of header, and it is encrypted and integrity protected. The MAC code, using SHA1, adds 20 bytes plus 15 bytes of maximum padding. The total overhead of the encrypted data is therefore 40 bytes, which is quite high only to send a small payload.

4.2.3 DTLS

The main problem with SSL/TLS is that the transport protocol used is TCP and it cannot be used with UDP, which it is a better option in order to have a lower delay in the exchange of the safety messages. For that reason, DTLS [42] can be used. DTLS is based on TLS and it provides the same security guarantees without the delays associated to stream protocols by using UDP. DTLS uses the same handshake as TLS providing the same communication privacy. The new problems DTLS has to solve [43] are packet loss, packet ordering, message size and DoS attacks:

- **Packet Loss**: DTLS uses UDP as transport protocol, so packet loss is a possibility. The way it is solved is by using a retransmission mechanism. Both client and server use a timer to retransmit the last packet sent. The retransmission is implemented using a state machine with 4 states: PREPARING, SENDING, WAITING and FINISHED. The timer length depends on the amount of retransmissions done, being the default length 1 second.

- **Message size**: An UDP datagram without IP fragmentation can only be 1500 bytes. The first issue is that the TLS handshake messages are bigger than 1500 bytes, so fragmentation is needed. DTLS solves that by using a fragment length and fragment offset field in the header. The problem of the message size when sending the safety data will be covered later.

- **Packet order**: in UDP pacekets can arrive in different order, so DTLS should be the protocol reordering the packets. It is solved by using an *epoch* and a *second sequence number* (or message sequence) to the header of a DTLS record layer. The epoch value starts at 0 and increases after every cipher state change, and helps to bond message received and encrypted packet's cipher. The sequence number is used to verify the MAC code. It starts at 0, increases with every message, and is reset when the epoch number is increased. The message sequence is used to combine the fragmented message.

- **DoS**: the only security requirement that is not solved in any of the solutions of the indirect communication is the availability of the system. Hardware problems aside, DoS attacks are the issues to solve. The DoS attack can be made in two sides: by several clients sending multiple handshake requests and slowing or denying the server capabilities; and

by amplification attacks. DTLS 1.2 adds a stateless cookie exchange in the handshake in order to prevent the DoS attacks. That solution adds new messages in the handshake, increasing the session creation time. That should not be a problem if the handshake is done before starting to send the safety messages.

By now, we see that DTLS solves the issues of using UDP in the handshake situation, plus it adds a DoS countermeasure. However it introduces a high overhead to the datagrams. Figure 4.1 shows that to send 6 bytes of data, DTLS forms a datagram of 35 bytes (29 bytes are overhead). That amount of bytes may be counterproductive, although now UDP is the transport protocol.

```
DTLS 1.2 Record Layer (35 bytes, 29 bytes overhead):
17 fe fd 00 01 00 00 00 00 00 05 00 16 00 01 00
00 00 00 00 05 ae a0 15 56 67 92 4d ff 8a 24 e4
cb 35 b9

Content type:
17
Version:
fe fd
Epoch:
00 01
Sequence number:
00 00 00 00 00 05
Length:
00 16
Nonce:
00 01 00 00 00 00 00 05
Ciphertext:
ae a0 15 56 67 92
ICV:
4d ff 8a 24 e4 cb 35 b9
```

Figure 4.1: DTLS overhead, as shown in RFC 7400 [44]

About the implementation in Android, no standard Java or Android implementation support it. There is an implementation of DTLS done by wolfSSL [45], but nevertheless that implementation is done in C. In order to set up a Java or Android environment, a JNI (Java Native Interface) should be implemented. By now some releases help the use of wolfSSL with a JNI, but still with several implementation problems.

4.3 Comparison Charts

Tables 4.1 and 4.2 are the ones showing the results of the analysis done towards securing communication protocols, both for direct and indirect communication. The first table is checking the security and privacy requirements, while

Protocol	**DSRC**	**CoAP**	**BLE**	**SSL/TLS**	**DTLS**	**IPSec**
Confidentiality	YES	YES	NO	YES	YES	YES
Authentication	PKI	PKI	PKI	PKI	PKI	YES
Integrity	PKI	PKI	PKI	YES	YES	YES
Non-Repudiation	PKI	NO	NO	PKI	PKI	NO
Availability	NO	NO	NO	NO	PKI	NO
Privacy	PKI	YES	YES	YES	PKI	NO

Table 4.1: Security & Privacy Requirements

Protocol	**DSRC**	**CoAP**	**BLE**	**SSL/TLS**	**DTLS**	**IPSec**
Hardware Changes	YES	NO	NO	NO	NO	NO
Java/Android Library	NO	NO	YES	YES	wolfSSL	LIMIT
Low Delay	YES	YES	NO	-	-	-
Overhead (Bytes)	-	-	-	15	29	57

Table 4.2: Implementation Requirements

the second checks the implementation (delay, need of hardware changes, and the software libraries) requirements of the solutions. This section is meant to choose one direct and one indirect protocol to use in the system architecture.

4.3.1 Meeting Security & Privacy

- *Direct communication*: V2V and M2M protocols provide confidentiality; although they presented problems with the privacy of the user; something that BLE solves. However, BLE does not ensure the confidentiality of the data. The rest of the requirements can be met with the use of a VPKI solution. Several works have been done in this area, for example in [26], [28], [46], [47], [48], and [49], although the discussion of how it can be implemented will be done in the following chapter.

- *Indirect communication*: From a first look, SSL/TLS and DTLS seem to be better options than IPSec, due they can solve almost all the requirements by only implementing a good PKI solution. The introduction of stateless cookies in the handshake by DTLS makes it a better option, due it creates a protection against DoS attacks.

4.3.2 Meeting Implementation Requirements

- *Direct communication*: from the information in table 4.2, DSRC and BLE are the best options due they provide low delay, with the advantage that BLE does not need hardware changes in vehicles or mobile phones. DSRC needs hardware or firmware changes in mobile phones in order to be able to receive and transmit in DSRC band, while BLE can be implemented as software in vehicles (either using a mobile phone with Bluetooth capability or in the OBUs).

- *Indirect communication*: The first thing to take into account is the overhead of the protocols. IPSec introduces around 37 bytes more than an usual IP Packet, so for a fast wireless transmission is not a good option. Between SSL/TLS and DTLS, the last one introduces 29 bytes of overhead against the 15 bytes of SSL/TLS. This extra bytes comes from all the values DTLS needs to deal with the packet loss and order. In addition, the handshake of DTLS has 2 extra messages (the stateless cookie), so in terms of transmission delay SSL/TLS is a better option. Library availability speaking, also SSL/TLS is on the lead due it is available in Java and Android.

As a conclusion, in the direct communication **DSRC** is the best option in terms of security and privacy. The protocol standard is already prepared for the situation of V2V communication, and although mobile phones cannot communicate in the safety channel, services channels can be used. Next chapter will discuss how DSRC can be implemented in mobile phones and how they can communicate with cars, although this document will not present an implementation test of the direct communication.

For the indirect communication a choice is quite more difficult. DTLS is a better option in terms of security and privacy, while SSL/TLS gives better performance and it is easy and reliable of implement. The proposed solution here is going to be like the architecture itself: an hybrid. The initial handshake is going to be done via **SSL/TLS**, where the nodes are going to authenticate themselves and exchange the cryptography information for the application data. Thanks to the use of a PKI/VPKI solution, all the security requirements are going to be ensured. An own application protocol running over UDP - **S-UDP** - is going to deal with the information exchange and the rest of the requirements.

Chapter 5

Communication Protocol

Chapter 3 presented an scheme of the hybrid communication between pedestrians and vehicles in order to exchange safety messages in pre-crash scenarios. From the conclusions taken in chapter 4, for the direct communication part DSRC seems to be the best option mainly due it is one of the standards for the V2V communication, so the pedestrians can be treated as another node of the VANET. Moreover, pedestrians will be able to send and also receive information from the vehicles. Chapter 4 concludes that in order to meet the security requirements for the communication using DSRC, a PKI (in that case, VPKI) solution is needed. Moreover, the implementation of the solution will need to make some changes in the firmware of the mobile phones in order to be able for them to work on the DSRC band using the Wi-Fi chipset.

For the indirect communication, SSL/TLS and S-UDP is going to be used. The first protocol will solve most of the security requirements while exchanging some shared data, to later on use UDP in a safety and private way in order to exchange the application data. Figure 5.1 shows the complete system architecture of the solution proposed.

5.1 Direct Communication

5.1.1 Firmware Changes

The DSRC band works in the range of 5.85 GHz to 5.925 GHz, and mobile phones nowadays do not support the communication with that band. Several solutions go through hardware changes like in [50], where the authors designed a RF front-end module (FEM) interfaced with a base-band processor on an

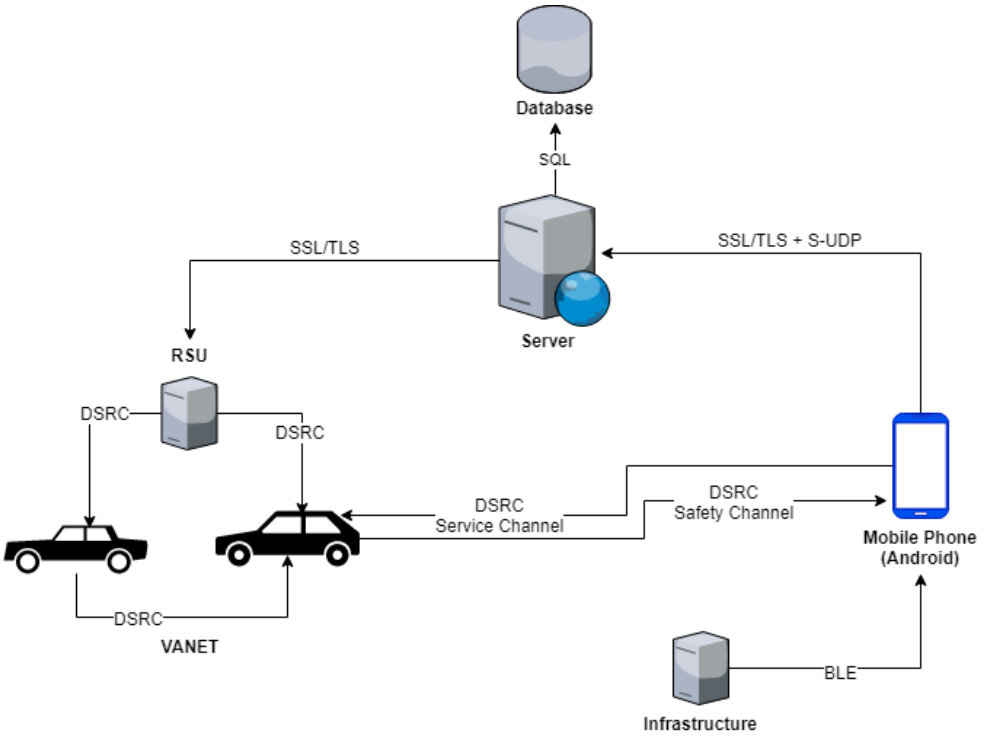

Figure 5.1: WARNING message creating process

FPGA and connected to Android phones. The authors concluded that the solution was able to communicate in the band of 802.11p, without being critical to the power consumption and the budget increment. Although the solution seems to work, extra hardware may lead in the disuse from the users. That is why the solution proposed in [24] is the one suggested in this document. The Wi-Fi chipsets in the mobile phones enables the communication in the 5 GHz band, close to the DSRC band. That is why some modifications are needed in the firmware and driver of the Wi-Fi chipset:

- Including of the DSRC band operation.
- Enabling the reception of broadcast packets.
- Processing the broadcast messages without the need of Wi-Fi association.

With that solution, mobile phones are able to transmit in the DSRC band and to receive the messages from the V2V safety channel of DSRC: 172 (5.855 to 5-865 GHz). But the mobile phones are not allowed to transmit in that

channel, only to receive. By now, mobile phones would be able to receive the beacons from the vehicles and calculate proximity to the cars, warning the pedestrians. However, the goal is that the pedestrians send warning messages to the vehicles.

DSRC-enabled mobile phones are allowed to transmit in service channels, being that the solution to enable the transmission of the warning messages by the pedestrians. Doing that, channel 172 is not affected and overloaded with the introduction of the pedestrian messages. In order to implement that solution, a service channel for the V2P communication should be enabled by the Federal Communications Commission. In summary, the direct communication between pedestrians and vehicles can work in the following way:

- Changes in the firmware of the Wi-Fi chipset in order to enable the communication using the DSRC-band, and the reception of the broadcast messages without Wi-Fi association.
- **Reception**: through the channel 172 of DSRC, used for the V2V safety service.
- **Transmission**: using another service channel for the V2P communication.

Due the difficulty to implement that solution in a real environment, without the proper nodes and tools to transmit in a DSRC service channel; the test of that part of the design is left to future work. Nevertheless, the key of the whole system is that the indirect communication can help the overload and the efficiency of the communication between the pedestrians and the vehicles, so that is why only the indirect communication will be tested and after that, a discussion of the feasibility of the whole architecture will be done.

5.1.2 VPKI solution

All the security requirements in DSRC pass by the use of a VPKI solution that enables the nodes to handle certificates in order to exchange messages in a private way. DSRC does not establish any solution in their standard, only the operation with private-public cryptography. The good thing is that several VPKI solutions have been presented, being the one presented in [46] the most complete one.

The authors present SECMACE, a VPKI system compatible with IEEE 1609.2 and ETSI standars specifications. The system allows privacy, prevents linking pseudonyms based on timing information, and offers protection against honest-but-curious entities. They demonstrate that their virtual machines can process the requests with low delay, and the privacy required is met not adding that much overhead.

The system also provides efficient revocation [51] and scales [52]. Based on the provided credentials, they obtain an efficient and DoS resilient V2X communication [53] [54] or resilient and privacy preserving peer-to-peer location based service [55] [56].

5.2 Indirect Communication

The indirect communication is going to use an hybrid solution: first, SSL/TLS is going to be used to open sessions between the pedestrians and the server. In the session establishment, several information is going to be exchanged to encrypt and generate MAC codes. After that, the data messages are going to be sent using UDP as transport protocol and S-UDP, an own application that is going to be presented later. With that, several situations are trying to be solved:

- Reducing the amount of messages sent by the pedestrians.
- Analyzing dangerous situations from medium and low warnings.
- Enabling a way to communicate mobile phones and vehicles using technologies already implemented, without needing changes in the network or the nodes.
- Exchanging data in a safe and private environment.

The first issue to solve is the need of a **PKI solution** in SSL/TLS. Its handshake deals with the authentication of the user and the server, and exchanges the data to use symmetric cryptography during the delivery of the application data (dealing with the confidentiality and integrity). In wired environments, the Certificate Management Protocol - CMP - supports the interactions between users and management entities, and deals with the user's registration information and the certificate revocation lists. The problem arises when using CMP in wireless environments due the lower transmission bandwidth, and

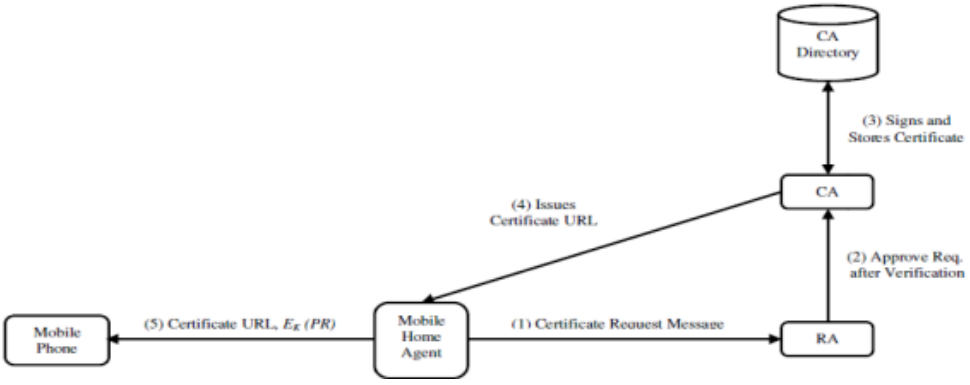

Figure 5.2: Certificate Management Operation [57]

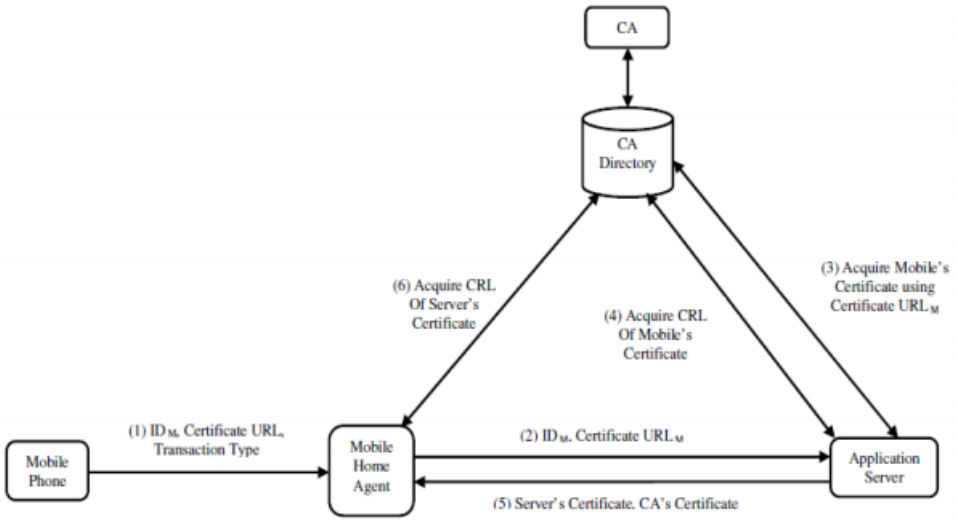

Figure 5.3: Certificate Management Validation [57]

the low computing power of mobile phones. Different solutions were proposed for mobile phones to use PKI solutions, mainly for e-commerce. It is interesting to merge both of the ideas, trying to overcome the limitations of the wireless scenarios:

- *Certificate URLs*: in [58], the authors proposed a solution using certificate URLs. The mobile phones do the requests to a Certificate Authority, which publishes the certificate and give a certificate URL to the user.
- *External Agent*: in [57] a Mobile Home Agent is proposed. That external agent is the one in charge of handling the mobile phones' information, and performing all the operations with the CA. Figures 5.2 and 5.3 show the different procedures (certificate management and validation) involving the Mobile Home Agent.
- *Temporal Certifications*: the solution allows to obtain several certificates by the Mobile Home Agent, and then giving them to the user once at a time; allowing the mobile phone to use a different certification each time it starts the service.

An alternative is to extend cellular authentication infrastructures connecting to V2X, e.g. as the work done in [59] and [60]. However, one interesting solution can be to adapt VPKI solutions to the Internet domain; so both vehicles and pedestrians can use the same solution, and pedestrians won't use two different PKI solutions depending if the communication is with a server (indirect) or with the vehicles (direct).

SEROSA [61] presents a privacy-preserving service-oriented architecture that merges Web Services and Vehicular credential management entities. Through extensive experimental evaluations, the authors demonstrate its dependability and efficiency compared to state of the art VPKIs.

5.2.1 SSL/TLS handshake

The SSL/TLS handshake will share by itself enough information for the nodes to exchange data in a safety way. The problem is that any application data sent after the handshake is done via TCP, add it will add some delay in the transmission. Also, the overhead SSL/TLS adds to the packets can be counterproductive.

That is why after a successful handshake, the client is going to send a HELLO message, answered with an OK/NOK from the server. Figure 5.4 shows the whole communication via SSL/TLS. Once the handshake is done, the client and server are exchanging data the following way. All the messages sent by now are encrypted using symmetric encryption, so that information exchanged is considered safe. Figure 5.5 shows the format of the messages sent after the SSL/TLS handshake.

- **HELLO**: the client is sending that message. It is sending the key to use to encrypt/decrypt and generate MAC messages. The algorithms the application uses are the following:
  - **Encryption**:
    - *AES*: the encryption algorithm used is the Advanced Encryption Standard, that uses keys of 128, 192 and 256 bits, and a 16 byte block size. The solution proposed is going to use the 256 bits option.
    - *CBC*: the Cipher Block Chaining is the block cipher mode of operation. In CBC, each block of plaintext is XORed with the previous ciphertext block before the encryption. To have a even more unique message, an initialization vector (IV) must be used in the first block. The IV is going to be sent by the server.
  - **MAC Algorithms**: the algorithm used is going to be *SHA-256*. It generates an almost-unique, 32-byte hash. The key size can be any length, although the recommended size is 64 bytes. Due the key exchanged is 32 bytes, another 32 bytes are padded.
- **OK**: if everything goes fine, the server is sending the session ID of the client, and the IV used to encrypt and decrypt the messages.
- **NOK**: if somethings goes wrong, the server will send a NOK message so the client should try to send a HELLO message again.

5.2.2 S-UDP: WARNING messages

The SSL/TLS handshake and data exchanged is used to establish certain shared information between the pedestrians and the server. It may seems quite redundant to use a SSL/TLS communication to exchanged cipher data, when the

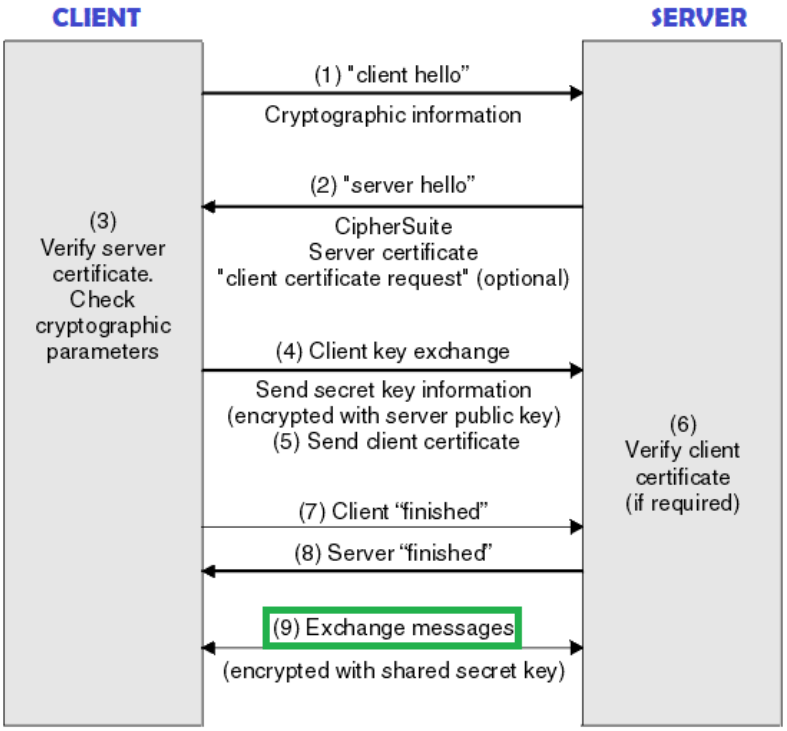

Figure 5.4: TLS handshake

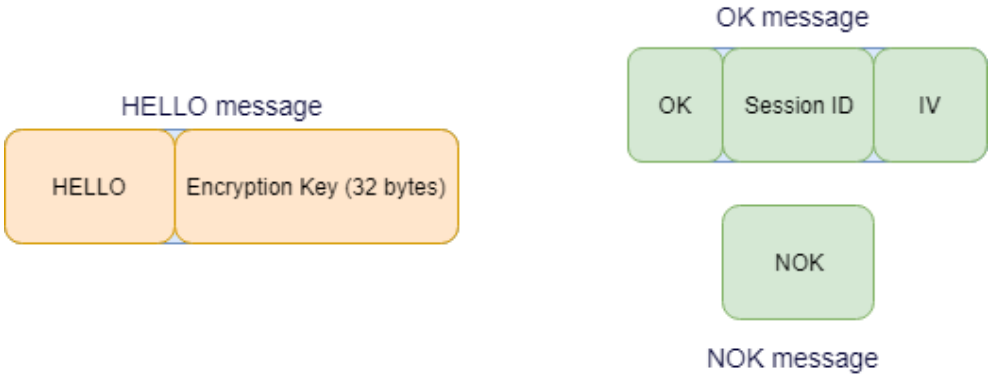

Figure 5.5: Messages exchanged during SSL/TLS session

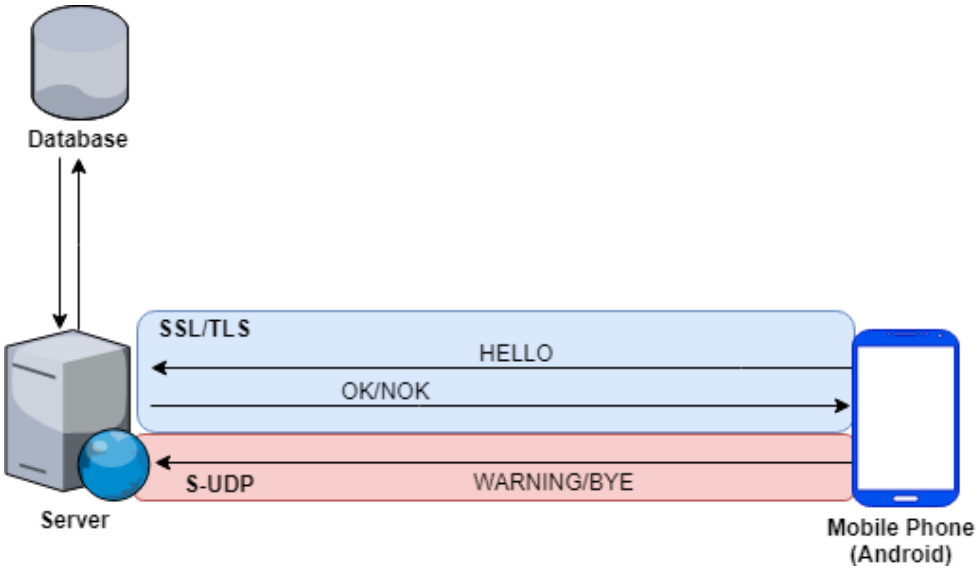

Figure 5.6: Android - Server schema

handshake does that. However, the delivery of safety messages is going to be done using UDP datagrams, from scratch unsafe, so some shared information is needed to encrypt and sign those messages. Figure 5.6 shows the complete communication done between the pedestrians and the server, with all the messages exchanged.

The schema shows both the SSL/TLS connection, and the S-UDP one. 2 messages are exchanged via UDP datagrams: WARNING and BYE. The first one are the warning messages the server is going to process to later inform the vehicles. The second message (BYE), is used to tell the server to remove the session ID of the user, and therefore the session. The BYE message is going to be sent when the user is changing mode of operation, or when the application is closed. The format of the messages is shown in figure 5.8, and the information they exchange is the following:

- *Session ID*: to identify that the user has an open session, so it was authenticated before. Also, the server will link the ID to the keys in order to check the integrity of the message, and decrytp it.
- *Mode of operation*: between walking, running or biking. The server will need that information in order to be able to analyze better if a dangerous situation is taking place.
- *Level of warning*: the highest warning (level 3) is handled by sending a

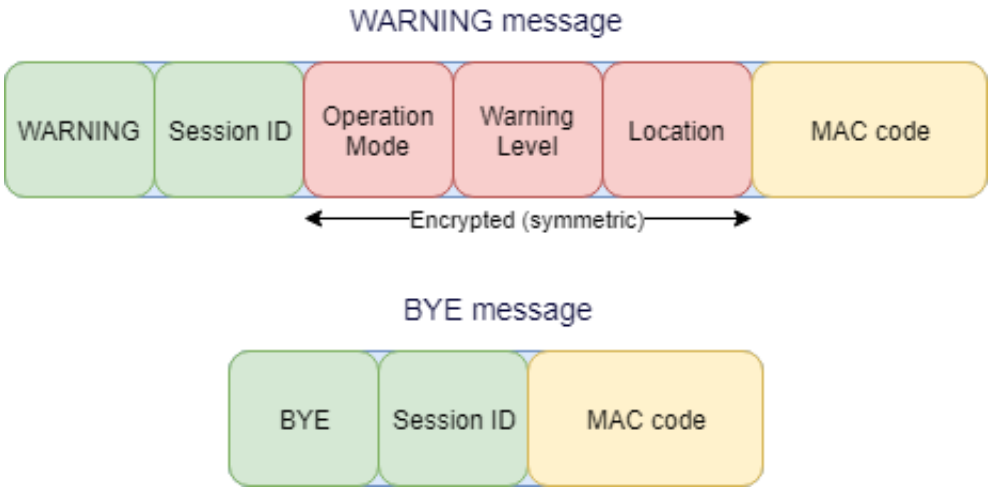

Figure 5.7: Warning Message

direct message to the vehicles, so only levels 1 and 2 will be notified via an indirect message.

- *Location*: obscured longitude and latitude of the pedestrian. The issue is to locate the user in the right area zone, so the exact location is not needed.

The confidentiality and integrity of the messages is ensured by the use of symmetric and MAC cryptography. On the other hand, the authentication is linked to the session ID: the SSL/TLS communication establishes the session ID, and the keys used for the S-UDP communication. So the PKI solution that handles the certificates used in the SSL/TLS handshake ensures the authentication of the user, in other terms, that the user is legitimate to send messages. The privacy of the user is ensure due:

- The location sent is obscured, so the server will not obtained the real location of the user.
- The information sent in the WARNING messages may link a session ID with the location of the user, although if the pedestrian is not generating those messages in a high frequency (like a beacon), an attacker that may read the messages cannot track the user.
- The location is linked to a session ID, but the session ID is not linked to the certificate used, so the location cannot be linked to a specific user.

5.2.3 Server Functionality

The processing unit is composed by two systems: a server and a database. The server is the one in charge of creating sessions and process the warning messages received by the trusted users, while the database is going to store the information of the sessions open and the data from the RSU connected to the service. The functionality of both the server and database is shown in figure 5.8. The SSL/TLS server will receive the requirements of new users, and the ones with a wrong certificate will be rejected. The UDP server will be reading all the datagrams sent, rejecting the ones with a wrong session ID, introducing a countermeasure for DoS attacks. The SSL/TLS server can be attacked by DoS, being that the only single point of failure of the processing unit.

Talking now about the reception of correct datagrams; once the authentication and the integrity is checked, and the information is decrypted, the information (mode of operation, level of warning and location) is sent to the application server. The application server will check in which zone the user is operating, by asking that information to the DB from the location. Once the zone is known, a relation between the mode of operation and the level of warning is done. Doing that, the server checks the degree of danger by checking the amount of users are doing a certain level of warning, for example:

- Supposing a walking mode of operation and level 1 of warning, the number of users doing that warning should be very high. A walking level 1 means that the user is looking at the phone or changing the way she moves. Only if a high number of users are doing that (for example in zones close to a football stadium) is where the system should notify the VANET.

- More serious cases, like a running level 2 warning, means that the pedestrians are trying to cross the road with a certain speed or they are running while looking at the phone. Now the number of users doing that should be lower.

- The worst situation is biking level 2 warnings, here the server will generate a warning message with a low number of pedestrians.

This relation will be checked by different counters linked to the RSU that they will be initiated every time the first message arrives. Each counter has a timer of 1.5 seconds, which is the window where the server is checking the amount of users generating the warning messages. Once the timer expires, the

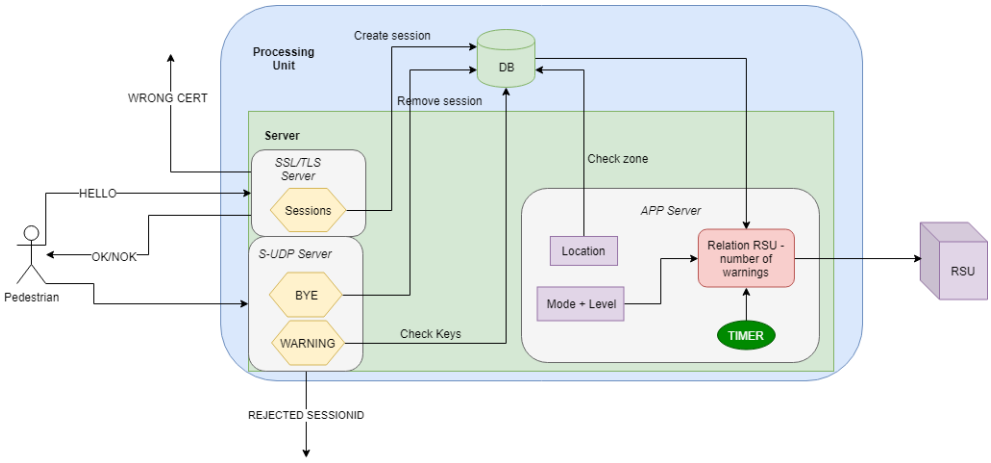

Figure 5.8: Processing Unit Functionality

counter is removed. The database is needed mainly to store the information of the different RSUs that are in the system. The database is composed only for 1 table, containing the following columns:

- RSU code, internal code used to start the counters in the application server.
- Latitude & Longitude, telling the location of the RSU.
- IP Address, of the RSU where the server is going to send the warning message.
- Distance, column used to compute the distance and choose the closest one to the location compared. The way to compute the distance is by using the Spherical Law of Cosines. Using that solutions gives really good results down to distances as small as a few meters - exactly want we want. The formula [62] is the following; where α is the latitude in radians, λ is the longitude in radians, and R is Earth's radius (can be in metres or kilometers depending on the final result we want):

$$d = acos(sin\alpha_1 \cdot sin\alpha_2 + cos\alpha_1 \cdot cos\alpha_2 \cdot \Delta\lambda) \cdot R \tag{5.1}$$

The language used to communicate the server and the database is SQL - Structured Query Language. SQL statements are used to perform several tasks, such as update data or retrieve data from a database. The different type of queries that can be used in SQL can be grouped as data manipulation, data

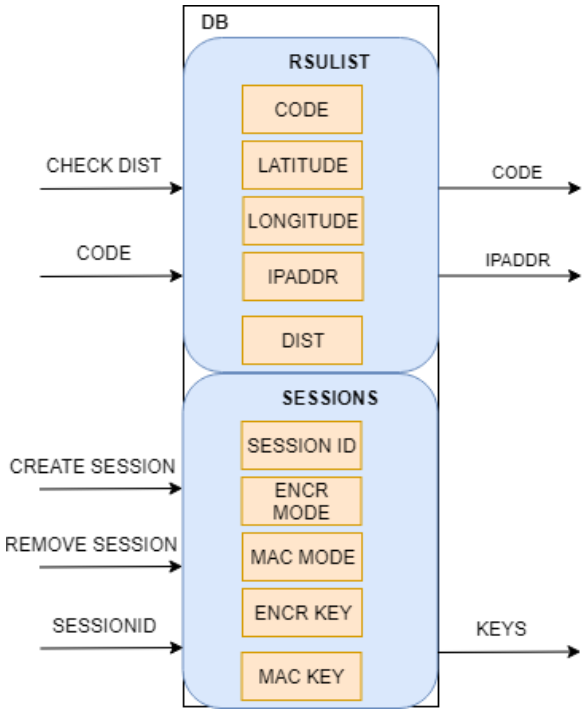

Figure 5.9: Inputs and outputs of the database

definition, and data access control. There are different management systems using SQL, all of them with their extensions only used in their systems. However, there are standard commands used in all of them, such as "Select", "Insert", "Update", "Delete", "Create" or "Drop". The queries used by the server in order to communicate with the database are the following ones:

- Select a row from a table using a constraint: SELECT $*$ FROM (table) WHERE (column) $=$ (value);
- Create a row: INSERT INTO (table) (columns) VALUES (values);
- Order the rows from one column and obtaining the first one: SELECT $*$ FROM (table) ORDER BY (column) ASC LIMIT 0,1;
- Remove a row: DELETE FROM (table) WHERE (column) $=$ (value);

From figure 5.9 it can be seen that the functionality of the database is simple, and that the actions done are not that much. Mainly the database is going to be used to check the code of the RSU from the location of the pedestrian, retrieve the IP Adress from the RSU Code, create and remove sessions, and obtain the keys from the session ID.

5.2.4 Server - RSUs

The last critical part of the whole indirect communication is the exchange of data between the server and the corresponding RSU, as shown in figure 5.10. Due the nodes in this section of the communication are fixed ones, the way to exchange information can be by just using trusted certificates. However DoS is a real issue that should be avoided, so SSL/TLS is going to be used. No S-UDP messages are sent, due no critical information is needed, the creation of a session will send a single WARNING message to notify the RSUs of a possible danger created by pedestrians in its zone.

Once the WARNING message is received and processed in the RSU, the way to notify the VANET can be done in two ways: by using the DSRC safety channel, or a service channel. Due there is no service channel available for V2P communication yet (as said in 5.1), use the safety channel can be a good option.

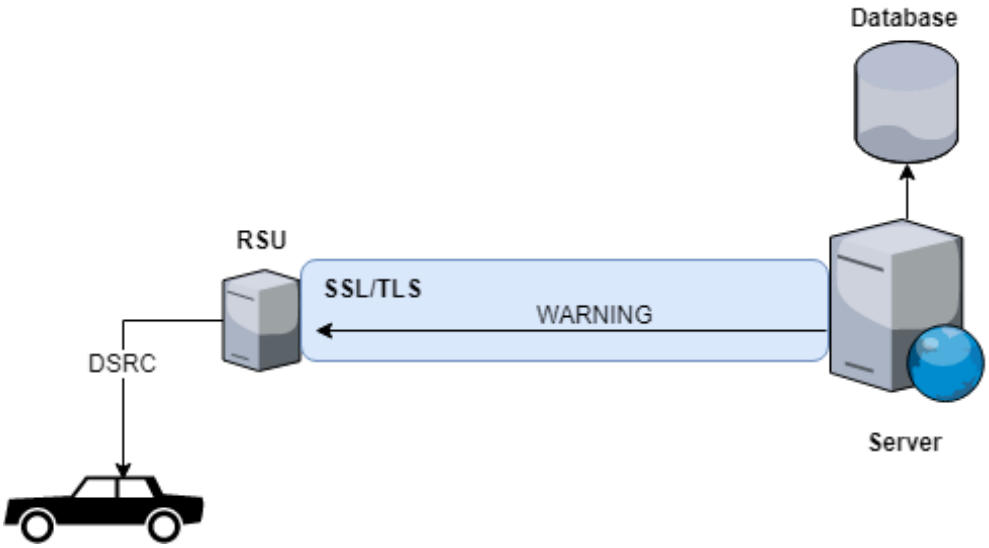

Figure 5.10: Server - RSU scheme

5.3 Revisiting Requirements

The solution has been presented, and now it is the time to check all the requirements analyzed in chapter 4. First let's talk about the implementation requirements: the communication should be achieved with the least delay as

possible (due the critical situation in pre-crash scenarios) and it has to be possible to implement in Android devices. The solution proposed for the direct communication was the use of DSRC, so the pedestrians can send direct messages to the vehicles in high-level warning situations. The use of DSRC allows the communication to be fast, and it can be achieved in mobile phones after a firmware change of the Wi-Fi chipset. The only constraint with that solution is the need to enable a service channel for the P2V communication.

The indirect communication uses the combination of SSL/TLS to start the service and exchange some data, to later use UDP (in a safe mode, S-UDP) to exchange the messages to the server. The use of datagrams enable the low delay in the exchange of messages between the pedestrians and the server, so then the server can quickly process all the data and determine when a warning message is needed to be sent to the corresponding RSU. Also, both SSL/TLS and UDP can be programmed with Android libraries, making easy and efficient its implementation.

About the security and privacy requirements, DSRC meets all of them except availability with the use of a proper VPKI solution. SECMACE is one of those VPKI solutions that can be implemented, following the specifications for V2X communications and removing the possibility of tracking due the use of pseudonyms based on timing information (protecting againts honest-but-curious nodes). About the indirect communication, SSL/TLS + S-UDP and the use of the server, the requirements should be analyzed more extensively:

- **SSL/TLS**: the security protocol is used to start the service between the pedestrians and the server. Both entities should use certificates in order to authenticate themselves, so in the first stage of the communication the *authentication* of the users is achieved. Besides, cryptography data is used in order to secure the messages exchanged using UDP, having the *confidentiality and integrity* of the information covered.

- **PKI solution**: the solution presented moves the processing of the certifications (obtaining and checking) in the mobile phones, and uses external agents to free resources in the devices. Also, it assures the requirements solved by SSL/TLS reducing the vulnerabilities of the basic use of certificates.

- **Server functionality**: it must be assumed that the server acts honestly, and that it will not work as honest-but-curious node. From that assump-

tion, the server will not relate the certificate use in the SSL/TLS transaction, with the session ID used in the UDP messages; so the *privacy* of the user is met. About the availability of the system, only the UDP port has a mechanism to avoid DoS attacks.

- **Vulnerabilities**: two are the main vulnerabilities that need to be solved. First the SSL/TLS can be attacked with DoS attacks, so a mechanism to avoid that should be implemented (can be based on the one that DTLS use). Also if a trusted user is misbehaving the service can stop its session ID, but its certificate won't be revoked; enabling that user to starts the service again. It will be blocked again when it starts to send wrong messages, but it is needed a mechanisms that allow to connect the session ID with the certificate of the user without linking its behaviour.

Chapter 6

System Evaluation

Although the design proposed covers direct and indirect communication, the implementation and evaluation done so far focuses in the pedestrian - server side. The full implementation and evaluation is left for future work. However, a successful implementation in terms of delay, warning detection, and security and privacy communication in the pedestrian - server side can be extrapolated to the final part of the system. The system evaluation analyzes the different stages from the detection of dangerous movement of the user to the moment the server determines a dangerous situation:

1. **Decision algorithm**: the evaluation in that part will be how good the detection of dangerous situations is done. A test user will perform several actions near the road, and the expected and obtained result will be compared.

2. **Secure and private communication**: the analysis of this stage will be done observing the packets sent by the mobile phone to the server, and analyzing them in terms of delay and security efficiency. The requirements to determine that the communication works will be checked.

3. **Server functionality**: the analysis will determine if the server is successful in the reception of the datagrams from the mobile phone, and processing the information in order to determine correctly when a warning message is needed to be sent, to the proper RSU.

6.1 Decision Algorithm

The way to test the decision algorithm was done in two different ways. In 2.4 the decision algorithm is presented, and two different situations trigger the algorithm to work and decide the level of warning: big changes in the acceleration of the pedestrian, and the proximity to road. Both of them can happen at the same time, but this won't change the final result. Therefore, separate tests were done in order to determine if the result given by the algorithm meet with the criteria followed:

Mode	**Situation**	**Value Expected**	**Value Obtained**
Walking	Running	1	1
	Gradually acc.	2	1
	Looking phone	2	2
	Run & Phone	3	3
Running	Walking	0	1
	Faster	1	1
	Weird movements	2	2
	Look phone	3	3
Biking	Faster	2	1
	Looking phone	3	3

Table 6.1: Change of acceleration situations' tests

- *Change of acceleration*: a pedestrian doing dangerous movements in urban and suburban areas, with normal and faster speeds - depending on the mode. 10 different situations were tested: 4 while walking, 4 while running, and 2 while biking.
- *Proximity to road*: another mobile phone is used as "beacon device" sending beacons every 2 seconds. The pedestrian, with another mobile phone in the pocket, walks next to the mobile phone doing weird movements and looking at the phone. 8 situations were tested: 4 while working and 4 while running (while biking the proximity to road is not applicable).
- All the tests were done during daylight, due that the detection of night would only sum 1 in the warning detection.
- Each situation was tested 5 times, and in tables 6.1 and 6.2 the result shown is the mean of all of them.

Mode	Situation	Value Expected	Value Obtained
Walking	Running	1	1
	Gradually acc.	2	1
	Looking phone	2	2
	Run & Phone	3	3
Running	Walking	0	1
	Faster	1	1
	Weird movements	2	2
	Look phone	3	3

Table 6.2: Proximity to road situations' tests

From the results in tables 6.1 and 6.2 we can see how the obtained results are really close to the ones expected. The situations were the expectations are different from the real data are where the speed is changing in a gradual way. That happens most likely due to the delay in the GPS information, so only when the speed is below or above the limits during enough time, it is when the algorithm can detect the change. Aside from that, the results are really promising in detecting when the user is creating dangerous situations in the road.

6.2 Communication & Server Functionality

Once the decision algorithm has been tested, obtaining very promising results, it is time to test the communication between the Android mobile phones and the server. As said at the beginning of this chapter, the implementation presented alongside with the full design is the connection Android - server and the processing of the data. The implementation and test of the last part of the communication with the VANETs can be left for future work. All the code and the presentation of the Android application will be explained in Annex A. The test was performed as follows:

- Three mobile phones, with the Android application, sent several messages to the server. In order to test all the possible situations in the road, the version of the application in those mobile phones had a test button generating random WARNING messages. Two of these mobile phones sent correct messages to the server, while the other mobile phone sent wrong messages (all of the phones are treated as trusted users).

- The time of the test was 5 minutes. The amount of correct messages was higher than wrong messages, in a percentage of 70/30.
- The server was built in a laptop, accepting the SSL/TLS and UDP requests in ports 9999 and 8888, respectively. The IDE used to build the server was NetBeans, using Java DB as database.
- In order to test the functionality of the server checking the nearest RSU, three random locations were introduced in the database in order to simulate the location of three RSUs. Figure **??** shows the map of the area in the Stockholm city with the locations introduced:
  - **STOCK01**. Location: 59.347853, 18.070611.
  - **STADION02**. Location: 59.343134, 18.082173.
  - **GARDET03**. Location: 59.348621, 18.102439.
- All the test were performed in a private network near the location of STOCK01. Both mobile phones sent several test messages in order to check how those messages were sent (to test the security and privacy of the users) and how the server processed those messages (to test the server functionality).

A total of 70 messages were sent in 5 minutes, being 49 of them correct ones and 21 wrong ones. The following conclusions can be made after analyzing the data obtained:

- All the wrong messages were blocked. From those messages blocked, the 25% (5) were using a wrong session ID. 16 messages were impersonation a trusted user, and all of them were blocked during the MAC code checking.
- From the 49 correct messages, 46 were processed correctly (what makes a 95% of success). The 3 correct messages were not processed due a problem of the server code, so if two messages were received at the same time but one of them was a wrong one, the correct one was also blocked.

Once the whole performance of the communication pedestrians-server is checked, let's analyze the security requirements of the messages exchanged, as talked in chapter 3. One of the correct transactions is used in order to analyze the SSL/TLS transaction, and the UDP messages sent. Figure 6.1 shows a

whole communication between the pedestrian and the server from the server's point of view. About the user creation, we can only see how the "HELLO" messages is received, and an "OK" is sent. Later on, the packets exchanged will be analyzed. The UDP message reception is more interesting: the red box shows the encrypted information, the blue box is showing the MAC code, and finally the green box is showing that the information is encrypted and analyzed by the server.

```
run:
SSL Server started in port: 9999
UDP Server started in port: 8888
New SSL/TLS connection
Message received: HELLO%AAkMFBoLep0zaIkWEozUuQ
Create new session with ID: fz0PqZLJXH
Answer Sent: OK%fz0PqZLJXH%jYDJnxXN+MjEY5Ya7GB27Q

Message Received: WARNINGSEPfz0PqZLJXHSEP/tNFqbY34IjFjqFwslOoFL2a/RxvwxEfBKCV0HnNMTrgBXHgZTWwTyqXk/Mo7ZVfSEPP5SHw0EmqcR/6t6KTM1zPnIBlgrmvAzldJJMwBM0Hl0
Decrypted Info: WalkingSEP1SEP59.3474802SEP18.0706991
The user fz0PqZLJXH in zone STOCK01 reported a warning with mode Walking and level 1
```

Figure 6.1: SSL/TLS and UDP message reception in the server

All the exchanged was tracked with the use of Wireshark, and the SSL/TLS communication is shown in figure 6.2. After the communication is started by the mobile phone (sender), both sender and receiver (server) are exchaging certificates and ciphers. After the server is sending the message "Change Cipher Spec", the common data is shared so both nodes can proceed to send messages. Another two messages can be seen labeled as "Application Data": the HELLO message from the mobile phone, and the corresponding OK from the server.

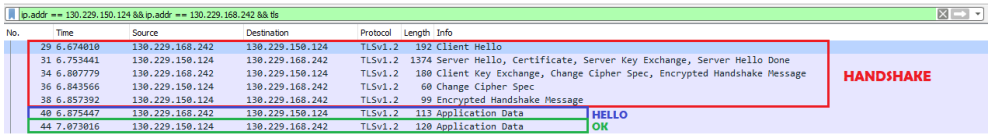

Figure 6.2: Wireshark: TLS communication

In the TLS/SSL communication, there is no constraint with the delay due the safety information is intended to be sent later on. The constraints in that part of the communication is the security and the privacy. Supposing a good PKI solution is implemented, and both client and server has a trustful certificate, figure 6.3 shows how both client and server sent their public keys to the other node after checking the certificates. It can be said that the authentication of the users, and the integrity and confidentiality of the information exchanged is ensured; so the keys exchanged to secure the UDP data is safe.

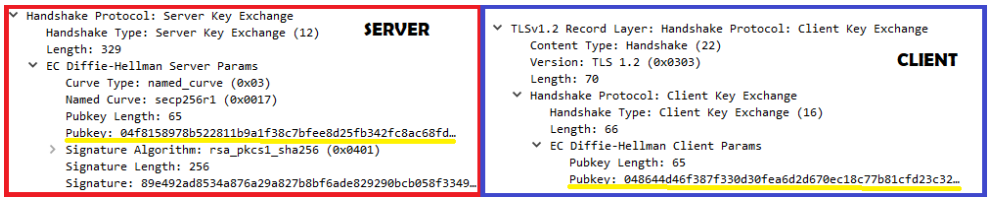

Figure 6.3: Wireshark: TLS communication

Let's move now to the UDP exchange. Figure 6.4 shows an example of one of the safety messages exchanged. The requirements we need to check are the security of the information exchanged and the delay:

- **Message Delay**: the data exchanged (useful data) is 102 bytes, and the whole packet is 144 bytes. The overhead of the message is 22 bytes, which enables to send the required information without adding to many bytes. In a 802.11g network, with a data rate of 54 Mbps, the propagation time was 21 μs (sending the package to the access point, and way back to the server).

- **Security**: as shown in figure 6.4, the application data shown in wireshark is illegible, due it is composed by the encrypted data and the mac code.

From the things checked, we can say that the communication between the mobile phones and the server meets all the requirements desired, with some exceptions: there is no mechanism of avoiding DoS attacks in the TLS/SSL server, and no revocation list was checked in order to avoid bad behaviour from trusted users.

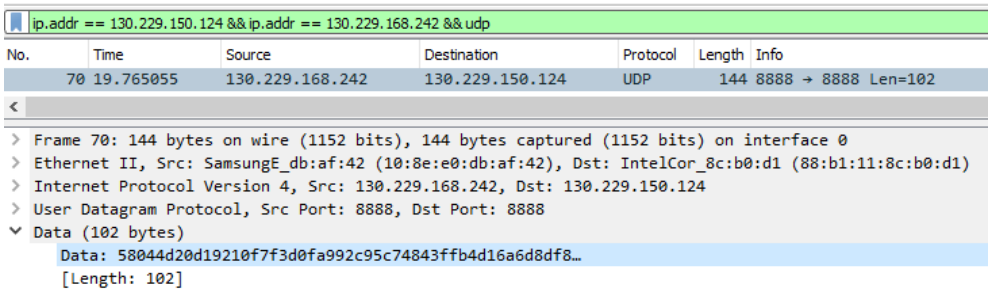

Figure 6.4: Wireshark: UDP communication

Chapter 7

Conclusions

This document presents a solution architecture that tries to introduce the pedestrians into the V2V communication (V2P) in a safe and private way, without losing the efficiency that V2V communication has. The first problem presented is what information pedestrians are going to send to vehicles. In other words, how to identify when a pedestrian is creating a danger that vehicles in the area need to know. Solutions in the market use the camera and the microphone of the mobile phones in order to detect when a user is close to a car or the road. These solutions are really good due the information they provide is very precise. However, they introduce several issues like the privacy of the user, the huge processing times or the need to have always the phone in the hands, between other.

The solution proposed in this document tries to analyze the behavior of the pedestrian and determine, using several rules, if a dangerous situation is happening or it is very likely that it can happen. Using the built-in sensors of the phone, the location services, and the information from beacons in the roadside; an algorithm in the mobile phone tries to analyze if a warning should be sent to the cars nearby, and which level of warning should be. The solution provides a way to analyze misbehavior in the road, while protecting the privacy of the pedestrian. The results in 6.1 were very promising, giving nice results between the levels of warning given by the algorithm and the expectations from the rules proposed. Of course, there is room for improvement in that sense, and it will be analyzed later on.

With the warning algorithm defined, the next thing to solve is how to send that information from the pedestrians to the vehicles. The architecture pro-

posed is an hybrid schema, in other terms, depending on the type of warning that the pedestrian is generating, the message sent to the VANET is going to be direct or indirect. The novel ideas are that the pedestrians are the ones alerting the vehicles (and not backwards, as most solutions do), and trying to reduce the amount of messages pedestrians will send (by processing some of them).

Talking more in detail about the indirect communication, that server is the one processing most of the warnings received. As a result, only an accumulation of small warnings (either by time or by number of people) is going to be reported to the vehicles. The tests done to the server functionality gave nice results: the server was able to identify the nearest RSU from the location and the processing of the messages received from the pedestrians followed the rules defined. As it happened with the definition of the rules, the criteria followed are assumptions done by the author, so there is room for improvement. Nevertheless, the solution works properly and the server reception and send of warning messages follow the expected.

Having all the nodes of the system defined, and how the pedestrian information is obtained and treated, the last part in the solution proposed is the choice of the correct communication protocol for each type of communication. First, several requirements (security, privacy and implementation) were defined based on the architecture and the scenario to solve (pre-crash). Several protocols can be implemented in order to provide security to the communication between the pedestrians and the server, and pedestrians and vehicles. For the direct communication DSRC, CoAP and Bluetooth were analyzed; and SSL/TLS, DTLS and IPSec for the indirect communication. After comparing all the results obtained, DSRC was chosen to be used in the direct communication, and an hybrid between SSL/TLS and UDP (secured with shared keys) for the indirect communication. The full communication protocol was presented merging the hybrid architecture and both protocols, presenting as well the application protocol used to communicate pedestrian and server, and the server functionality.

Section 6.2 shows the tests done in order to check the server functionality and the requirements of the system. The SSL/TLS handshake ensures the authentication of the users involved in the communication, as well as secured the shared information needed in the datagram sent. The analysis of the packets exchanged during the handshake showed that all the data exchanged expected was done, and in a safe way. In addition, the application data sent was en-

crypted and integrity was ensured as expected. In terms of delay, the delivery of UDP protocols enable to send the warning message with the least transmission time possible. Only two issues need to be solved: a protection mechanism for DoS attacks in the SSL/TLS server, and a way to avoid the misbehavior of users. The misbehavior detection requires its own mechanisms, along the lines of works such as [63], [64], [65], and [66]

In summary, the system gives promising results when analyzing the pedestrian road data and sending it from the mobiles phones to the server. Although the document presents a full architecture from the pedestrians to the vehicles, the implementation is not completed due the constraints in the DSRC communication or the lack of some test nodes. A full implementation will be needed in the future in order to test the full system but nevertheless, the tests done leave good feelings to think that the final result will be successful.

Future Work

1. **Full implementation of the system**: to complete and test the communication between the pedestrians and the VANETs. In order to complete the implementation, and to be able to test it in real environments, several issues should be solved:

    - For the direct communication part, firmware changes should be done in the mobile phones in order to check the correct reception of safety beacons from the vehicles. The transmission part should be solved by using a DSRC service channel.
    - The server should be open to the public Internet, so the mobile phones can access from any network, without the constraint of having the communication in private networks.
    - Trustful certificates (PKI) should be used; the ones implemented in the solutions proposed in this document used a own certificate.
    - Real RSUs should be used, with a list of locations and IP addresses, so the server can use its functionality. Those RSUs should be located with enough separation in order to be able to test if the server can obtain the corresponding RSU from the pedestrian's location.

2. **Cloud computing**: the server exposes a single point of failure in the system, as well as it may happen that a big number of users using the service may result in a blocking in the server side. The Cloud offers the

possibility to process that information in a more dynamic way, and also it can be used to implement the translation of technologies and protocols. The Cloud functionality may provide both solutions at the same time. The new implementation can be the following one:

- The Android applications send the messages to the Cloud with the location and the raw data from sensors (removing the processing in the mobile phones). All the processing is done in the Cloud services, determining when a dangerous situation may happen in a certain area.
- The communication with the VANETs can be done or via direct messages from the Cloud services, or via requests from the RSUs every a fixed interval.
- Having the Cloud services as translators of protocols, the translation from V2V to Internet protocols can be done as well, so vehicles can send messages to the mobile phones. Those warning messages can be translated in notifications for the pedestrians, to warn them about dangerous situations in their area.

3. **Locating the pedestrian close to the road**: in the solution proposed, beacon devices are supposed to be in the roadside. That assumption is far from the reality, and not always can be done (for example, rural areas).

   - A way to determine if the user is approaching the roadside should be based on a efficient processing of the location information.
   - Some solutions used the location services and maps databases. AI can be a solution to implement in this area.

4. **Decision algorithm & Server rules**: the rules used both in the decision algorithm and in the server are based in the author's own rules. The definition of those rules may be different from another person, changing the overall performance of the system. Fuzzy mathematics can be implemented to create an AI system that determines when the user is going to generate a dangerous situation.

5. **Implementation of DTLS**: implementing the indirect communication using DTLS may help to check which solution is better. For that, wolfSSL library should be implemented correctly in Android.

Bibliography

[1] Y. Li, F. Xue, X. Fan, Z. Qu, and G. Zhou. "Pedestrian walking safety system based on smartphone built-in sensors". In: *IET Communications* 12.6 (2018), pp. 751–758. ISSN: 1751-8628. DOI: 10.1049/iet-com.2017.0502.

[2] Y. Tung and K. G. Shin. "Use of Phone Sensors to Enhance Distracted Pedestrians' Safety". In: *IEEE Transactions on Mobile Computing* 17.6 (June 2018), pp. 1469–1482. ISSN: 1536-1233.

[3] Tianyu Wang, Giuseppe Cardone, Antonio Corradi, Lorenzo Torresani, and Andrew T. Campbell. "WalkSafe: A pedestrian safety app for mobile phone users who walk and talk while crossing roads". In: (Feb. 2012). DOI: 10.1145/2162081.2162089.

[4] A. Hussein, F. García, J. M. Armingol, and C. Olaverri-Monreal. "P2V and V2P communication for Pedestrian warning on the basis of Autonomous Vehicles". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, pp. 2034–2039. DOI: 10.1109/ITSC.2016.7795885.

[5] Zhenyu Liu, Mengfei Wu, Konglin Zhu, and Lin Zhang. "SenSafe: A Smartphone-Based Traffic Safety Framework by Sensing Vehicle and Pedestrian Behaviors". In: *Mobile Information Systems* 2016 (Oct. 2016), pp. 1–13. DOI: 10.1155/2016/7967249.

[6] Myounggyu Won, Aawesh Man Shrestha, and Yongsoon Eun. *Enabling WiFi P2P-Based Pedestrian Safety App*. Apr. 2018.

[7] J. J. Anaya, P. Merdrignac, O. Shagdar, F. Nashashibi, and J. E. Naranjo. "Vehicle to pedestrian communications for protection of vulnerable road users". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. June 2014, pp. 1037–1042. DOI: 10.1109/IVS.2014.6856553.

[8] M. Bagheri, M. Siekkinen, and J. K. Nurminen. “Cellular-based vehicle to pedestrian (V2P) adaptive communication for collision avoidance”. In: *2014 International Conference on Connected Vehicles and Expo (ICCVE)*. Nov. 2014, pp. 450–456. DOI: 10.1109/ICCVE.2014.7297588.

[9] Pooya Rahimian, Elizabeth E. O’Neal, Shiwen Zhou, Jodie M. Plumert, and Joseph K. Kearney. “Harnessing Vehicle-to-Pedestrian (V2P) Communication Technology: Sending Traffic Warnings to Texting Pedestrians”. In: *Human Factors* 60.6 (2018), pp. 833–843. DOI: 10.1177/0018720818781365.

[10] StatCounter. *Mobile Operating System Market Share Worldwide, June 2018 - June 2019*. June 2019. URL: http://gs.statcounter.com/os-market-share/mobile/worldwide (visited on 07/11/2019).

[11] Android Developers. *Sensors Overview*. URL: https://developer.android.com/guide/topics/sensors/sensors_overview (visited on 04/18/2019).

[12] Android Open Source Project. *Sensor Types*. URL: https://source.android.com/devices/sensors/sensor-types (visited on 04/18/2019).

[13] Android Developers. *Motion Sensors*. URL: https://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-raw-data (visited on 04/18/2019).

[14] Android Developers. *Environment Sensors*. URL: https://developer.android.com/guide/topics/sensors/sensors_environment.html (visited on 04/18/2019).

[15] Android Developers. *Location Services*. URL: https://developer.android.com/guide/topics/location/strategies (visited on 04/18/2019).

[16] Android Developers. *Location*. URL: https://developer.android.com/reference/android/location/Location.html#getAccuracy() (visited on 05/12/2019).

[17] Shubham Jain, Carlo Borgiattino, Yanzhi Ren, Marco Gruteser, and Yingying Chen. "On the limits of positioning-based pedestrian risk awareness". English (US). In: *MARS 2014 - Proceedings of the 2014 Workshop for Mobile Augmented Reality and Robotic Technology-Based Systems, Co-located with MobiSys 2014*. Association for Computing Machinery, Jan. 2014, pp. 23–28. ISBN: 9781450328241. DOI: https://doi.org/10.1145/2609829.2609834.

[18] Android Developers. *Bluetooth Low Energy Overview*. URL: https://developer.android.com/guide/topics/connectivity/bluetooth-le (visited on 05/13/2019).

[19] AltBeacon. *Android Beacon Library*. URL: https://github.com/AltBeacon/android-beacon-library/issues (visited on 05/13/2019).

[20] AltBeacon. *AltBeacon*. URL: https://altbeacon.org/ (visited on 05/13/2019).

[21] Wikimedia Foundation. *Lux*. URL: https://en.wikipedia.org/wiki/Lux (visited on 04/18/2019).

[22] Parag Sewalkar and Jochen Seitz. "Vehicle-to-Pedestrian Communication for Vulnerable Road Users: Survey, Design Considerations, and Challenges". In: *Sensors* 19 (Jan. 2019). DOI: 10.3390/s19020358.

[23] Panos Papadimitratos, A. de La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza. "Vehicular Communication Systems: Enabling Technologies, Applications, and Future Outlook on Intelligent Transportation". In: *IEEE Communications Magazine* 47.11 (Nov. 2009), pp. 84–95. ISSN: 0163-6804. DOI: 10.1109/MCOM.2009.5307471.

[24] X. Wu, R. Miucic, S. Yang, S. Al-Stouhi, J. Misener, S. Bai, and W. Chan. "Cars Talk to Phones: A DSRC Based Vehicle-Pedestrian Safety System". In: *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*. Sept. 2014, pp. 1–7. DOI: 10.1109/VTCFall.2014.6965898.

[25] Panos Papadimitratos, Virgil Gligor, and Jean-Pierre Hubaux. "Securing Vehicular Communications - Assumptions, Requirements, and Principles". In: *Workshop on Embedded Security in Cars (ESCAR)* (2006), pp. 5–14.

[26] Panos Papadimitratos, Lavente Buttyan, Tamás Holczer, Elmar Schoch, Julien Freudiger, Zhendong Ma, Frank Kargl, and Jean-Pierre Hubaux Antonio Kung. "Secure Vehicular Communication Systems: Design and Architecture". In: *IEEE Communications Magazine* 46.11 (Nov. 2008), pp. 100–109. ISSN: 0163-6804. DOI: 10.1109/MCOM.2008.4689252.

[27] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. "Design of 5.9 ghz dsrc-based vehicular safety communication". In: *IEEE Wireless Communications* 13.5 (Oct. 2006), pp. 36–43. ISSN: 1536-1284. DOI: 10.1109/WC-M.2006.250356.

[28] "IEEE Approved Draft Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages". In: *IEEE P1609.2/D12, November 2015* (Jan. 2016), pp. 1–241.

[29] C. Laurendeau and M. Barbeau. "Threats to security in DSRC/WAVE". In: Aug. 2006. DOI: 10.1007/11814764_22.

[30] Z. Shelby. "Constrained application protocol (CoAP". In: *RFC7252*. June 2014.

[31] R. A. Rahman and B. Shah. "Security analysis of IoT protocols: A focus in CoAP". In: *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*. Mar. 2016, pp. 1–7. DOI: 10.1109/ICBDSC.2016.7460363.

[32] O. Garcia-Morchon M. Brachmann and M. Kirsche. "Security for Practical CoAP Applications: Issues and Solution Approaches". In: *10th GI/ITH KuVS Fachgesprch Sensornetze (FGSN)*. 2011, pp. 1–7.

[33] *Bluetooth 4.0 Core Specification - GATT*. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification/ (visited on 08/06/2019).

[34] M. Bon. *A Basic Introduction to BLE Security*. Oct. 2016. URL: www.digikey.com/eewiki/display/Wireless/ (visited on 08/06/2019).

[35] S. Kent. "IP Authentication Header". In: *RFC 4302*. Dec. 2005. DOI: 10.17487/RFC4302.

[36] S. Kent and R. Atkinson. "IP Encapsulating Security Payload (ESP)". In: *RFC 2406*. Nov. 1998. DOI: 10.17487/RFC2406.

[37] S. Kent and K. Seo. "Security Architecture for the Internet Protocol". In: *RFC 4301*. Dec. 2005. DOI: 10.17487/RFC4301.

[38] D. Clark. *Vulnerabilities of IPSEC: A Discussion of Possible Weaknesses in IPSEC Implementation and Pro*. Mar. 2002. URL: https://www.sans.org/reading-room/whitepapers/vpns/paper/760 (visited on 08/10/2019).

[39] P. Karlton A. Freier and P. Kocher. "The Secure Sockets Layer (SSL) Protocol Version 3.0". In: *RFC 6101*. Aug. 2011. DOI: 10.17487/RFC6101.

[40] T. Dierks and E. Rescorla. "The Transport Layer Security (TLS) Protocol Version 1.2". In: *RFC 5246*. Aug. 2008. DOI: 10.17487/RFC5246.

[41] A. Prodromou. *TLS Security 6: Examples of TLS Vulnerabilities and Attacks*. Mar. 2019. URL: https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/ (visited on 08/10/2019).

[42] E. Rescorla and N. Modadugu. "Datagram Transport Layer Security Version 1.2". In: *RFC 6347*. Jan. 2012. DOI: 10.17487/RFC6347.

[43] N. van Drueten. *Security analysis of DTLS 1.2 implementations*. Jan. 2019. URL: https://www.cs.ru.nl/bachelors-theses/2018/Niels_Drueten___4496604___Security_analysis_of_DTLS_1_2_implementations.pdf (visited on 08/10/2019).

[44] C. Bormann. "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)". In: *RFC 7400*. Nov. 2014. DOI: 10.17487/RFC7400.

[45] *WolfSSL Manual*. 2019. URL: https://www.wolfssl.com/docs/wolfssl-manual/ (visited on 07/15/2019).

[46] M. Khodaei, H. Jin, and P. Papadimitratos. "SECMACE: Scalable and Robust Identity and Credential Management Infrastructure in Vehicular Communication Systems". In: *IEEE Transactions on Intelligent Transportation Systems* 19.5 (May 2018), pp. 1430–1444. ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2722688.

[47] F. Kargl, P. Papadimitratos, L. Buttyan, M. Müter, E. Schoch, B. Wiedersheim, T. V. Thong, G. Calandriello, A. Held, A. Kung, and Jean-Pierre Hubaux. "Secure Vehicular Communication Systems: Implementation, Performance, and Research Challenges". In: *IEEE Communications Magazine* 46.11 (Nov. 2008), pp. 110–118. ISSN: 0163-6804. DOI: 10.1109/MCOM.2008.4689253.

[48] Panagiotis Papadimitratos and Jean-Pierre Hubaux. “Report on the “Secure Vehicular Communications: Results and Challenges Ahead" Workshop”. In: *ACM SIGMOBILE Mobile Computing and Communications Review (ACM MC2R)* 12.2 (Apr. 2008), pp. 53–64. ISSN: 1559-1662. DOI: 10.1145/1394555.1394567.

[49] Panos Papadimitratos, Levente Buttyan, Jean-Pierre Hubaux, Frank Kargl, Antonio Kung, and Maxim Raya. “Architecture for Secure and Private Vehicular Communications”. In: *IEEE International Conference on ITS Telecommunications (IEEE ITST)*. June 2007, pp. 1–6. DOI: 10.1109/ITST.2007.4295890.

[50] P. Choi, J. Gao, N. Ramanathan, M. Mao, S. Xu, C. Boon, S. A. Fahmy, and L. Peh. “A case for leveraging 802.11p for direct phone-to-phone communications”. In: *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. Aug. 2014, pp. 207–212. DOI: 10.1145/2627369.2627644.

[51] Mohammad Khodaei and Panos Papadimitratos. “Efficient, Scalable, and Resilient Vehicle-Centric Certificate Revocation List Distribution in VANETs”. In: *Proceedings of the 11th ACM Conference on Security Privacy in Wireless and Mobile Networks*. WiSec ’18. Association for Computing Machinery, 2018, pp. 172–183. ISBN: 9781450357319. DOI: 10.1145/3212480.3212481. URL: https://doi.org/10.1145/3212480.3212481.

[52] Mohammad Khodaei, Hamid Noroozi, and Panos Papadimitratos. “Scaling Pseudonymous Authentication for Large Mobile Systems”. In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec ’19. Miami, Florida: Association for Computing Machinery, 2019, pp. 174–184. ISBN: 9781450367264. DOI: 10.1145/3317549.3323410. URL: https://doi.org/10.1145/3317549.3323410.

[53] Hongyu Jin and Panos Papadimitratos. “Proactive certificate validation for VANETs”. In: *2016 IEEE Vehicular Networking Conference (VNC)*. Dec. 2016, pp. 1–4. DOI: 10.1109/VNC.2016.7835974.

[54] Hongyu Jin and Panos Papadimitratos. “DoS-resilient Cooperative Beacon Verification for Vehicular Communication Systems”. In: 2019, p. 101775. DOI: 10.1016/j.adhoc.2018.10.003.

[55] Hongyu Jin and Panos Papadimitratos. “Resilient Privacy Protection for Location-Based Services through Decentralization”. In: *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec ’17. Boston, Massachusetts: Association for Computing Machinery, 2017, pp. 253–258. ISBN: 9781450350846. DOI: 10.1145/3098243.3098268. URL: https://doi.org/10.1145/3098243.3098268.

[56] Hongyu Jin and Panos Papadimitratos. “Resilient Privacy Protection for Location-Based Services through Decentralization”. In: *ACM Trans. Priv. Secur.* 22.4 (Dec. 2019). ISSN: 2471-2566. DOI: 10.1145/3319401. URL: https://doi.org/10.1145/3319401.

[57] S. Ray and G. P. Biswas. “Design of Mobile-PKI for using mobile phones in various applications”. In: *2011 International Conference on Recent Trends in Information Systems*. Dec. 2011, pp. 297–302. DOI: 10.1109/ReTIS.2011.6146885.

[58] J. Lee Y. Lee and J. Song. “Design and implementation of wireless PKI technology suitable for mobile phone in mobile-commerce”. In: *Computer Communications* 30 (Feb. 2007), pp. 893–903. DOI: 10.1016/j.comcom.2006.10.014.

[59] S. Gisdakis, V. Manolopoulos, S. Tao, A. Rusu, and P. Papadimitratos. “Secure and Privacy-Preserving Smartphone-Based Traffic Information Systems”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.3 (June 2015), pp. 1428–1438. ISSN: 1558-0016. DOI: 10.1109/TITS.2014.2369574.

[60] V. Manolopoulos, S. Tao, A. Rusu, and P. Papadimitratos. “HotMobile 2012 Demo: Smartphone-Based Traffic Information System for Sustainable Cities”. In: *ACM Mobile Computing and Communications Review (ACM MC2R)* 16.4 (2012), pp. 30–31. ISSN: 1559-1662. DOI: 10.1145/2436196.2436213. URL: https://doi.org/10.1145/2436196.2436213.

[61] Stylianos Gisdakis, Marcello Laganà, Thanassis Giannetsos, and Panos Papadimitratos. “SEROSA: SERvice oriented security architecture for Vehicular Communications”. In: Dec. 2013. DOI: 10.1109/VNC.2013.6737597.

[62] Wikimedia Foundation. *Spherical Law of Cosines*. URL: https://en.wikipedia.org/wiki/Spherical_law_of_cosines (visited on 05/07/2019).

[63] M. Raya, P. Papadimitratos, V. D. Gligor, and J. Hubaux. “On Data-Centric Trust Establishment in Ephemeral Ad Hoc Networks”. In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. Apr. 2008, pp. 1238–1246. DOI: 10.1109/INFOCOM.2008.180.

[64] M. Poturalski, P. Papadimitratos, and J. Hubaux. “Formal Analysis of Secure Neighbor Discovery in Wireless Networks”. In: *IEEE Transactions on Dependable and Secure Computing* 10.6 (Nov. 2013), pp. 355–367. ISSN: 2160-9209. DOI: 10.1109/TDSC.2013.17.

[65] S. Gisdakis, T. Giannetsos, and P. Papadimitratos. “Security, Privacy, and Incentive Provision for Mobile Crowd Sensing Systems”. In: *IEEE Internet of Things Journal* 3.5 (Oct. 2016), pp. 839–853. ISSN: 2372-2541. DOI: 10.1109/JIOT.2016.2560768.

[66] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. “SHIELD: A Data Verification Framework for Participatory Sensing Systems”. In: *Proceedings of the 8th ACM Conference on Security Privacy in Wireless and Mobile Networks*. WiSec ’15. New York, New York: Association for Computing Machinery, 2015. ISBN: 9781450336239. DOI: 10.1145/2766498.2766503. URL: https://doi.org/10.1145/2766498.2766503.

Appendix A

Android App and Java Code

The code of the implementation of the communication pedestrian - server can be found in the following links:

- Android APP: `https://github.com/Pablo19Sanchez/Android-v2p`
- Server: `https://github.com/Pablo19Sanchez/V2PServer`

A.1 Android Application

The code provided in the link above consists in two activities: in the first one (**FirstActivity**) the user will open a session with the server, obtaining the session ID and the keys needed to secure the communication. If that communication with the server is successful, and no errors occured, the user will be able to select the mode of operation and move to the following activity. The second activity (**modeActivity**) is the one doing all the work:

- Reading the information from the built-in sensors.
- Obtaining the information from the location services.
- Reading the beacons from the Bluetooth LE devices.
- Obtaining the level of warning and sending the datagrams to the server.

Apart from the two activities explained, several classes are used in order to implement the functionality of the application. Depending on the purpose, two groups can be made:

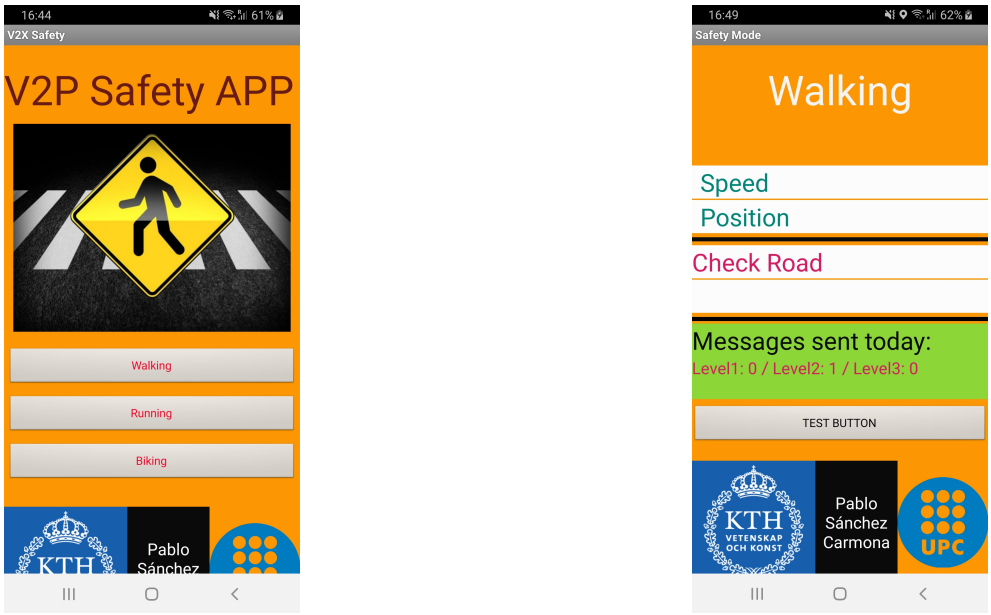

Figure A.1: Application View

- *Communication classes*: Android does not allow the use of communication methods in the activities, due connection errors may occur. For that reason, asynchronous tasks are used in order to connect to the server, send the messages, and receive messages (if applicable).
  - Classes **LoginRegistrationAsyncTask** and **SendDatagramAsyncTask** are the ones used to start the asynchronous tasks.
  - Classes **TCPClient** and **UDPClient** are used to handle data and methods intended to the communication itself.
- *Entities*: two entity classes are used in order to manage the application information.
  - **decisionClass** is the one processing the input information of the decision algorithm, and giving the level of warning to send to the server.
  - **MessageV2X** is the one creating the messages to send to the server: HELLO, BYE and WARNING.

A.2 Server Code

The code provide in the link above, the one of the server, has several classes intended to retrieve and process the data sent by the mobile phone applications:

- **V2XServer**: that is the main class, and opens the SSL/TLS and UDP ports. When receiving a message in one of the ports, it sends the information to the handler classes. When receiving the processed info about the warning messages, it saves those warnings until a message is needed to be sent to a corresponding RSU.
- **ClientHandler**: is the handler in charge to process the requests to open a new session. Checks that the user is legit, generates a session ID, and introduces in the database the info of the new session.
- **MessageHandler**: the one in charge of processing the info from the WARNING messages: checking the MAC codes, and decrypting the data. It sends the processed info about the warnings (sessionID, mode, and level of warning) to the main class. It also removes sessions when a BYE message is received.
- **MySQLData**: that class is the one creating the SQL queries and handling the information of the database.
- **RSU**: that class is an entity one, it stores the number of warnings depending on the mode of a certain area. After the number of warnings are lower than a certain number, it sends a WARNING message to the corresponding RSU.

TRITA-EECS-EX-2020:78
www.kth.se