Tunisian Republic

Ministry Of Higher Education And Scientific Research

Tunis El Manar University

Higher Institute Of Computer Science El Manar

Graduation Thesis

Submitted in accordance with the requirements for the :

National Engineering Degree in Applied Science and Technology

Speciality : Engineering and development of communication infrastructures and services

By

Oumayma BEN KHALIFA

Implementation Of Quantum SMC Use-Case for VANETs

Academic institution Supervisor(ISI-ElManar) : **Mrs Monia NAJAR**

Host institution Supervisor (It-Aveiro) : **Mr Armando Pinto**

RESEARCH LABORATORY : Optical Quantum Communications Group, IT -Instituto de Telecomunicações

Academic Year 2021 - 2022

Tunisian Republic

Ministry Of Higher Education And Scientific Research

Tunis El Manar University

Higher Institute Of Computer Science El Manar

Graduation Thesis

Submitted in accordance with the requirements for the :

National Engineering Degree in Applied Science and Technology

Speciality : Engineering and development of communication infrastructures and services

By

Oumayma BEN KHALIFA

Implementation Of Quantum SMC Use-Case for VANETs

Academic institution Supervisor(ISI-ElManar) : **Mrs Monia NAJAR**

Host institution Supervisor (It-Aveiro) : **Mr Armando Pinto**

RESEARCH LABORATORY : Optical Quantum Communications Group, IT -Instituto de Telecomunicações

Academic Year 2021 - 2022

I authorise the student to submit her internship report for the Internship Paper Defense-Oral Exam.

Host institution supervisor, **Mr Armando Pinto**

Signature & stamp

I authorise the student to submit her internship report for the Internship Paper Defense-Oral Exam.

Academic supervisor, **Mrs Monia NAJAR**

Signature

Acknowledgments

First and foremost, I would like to thank my first supervisor, Professor **Armando PINTO**, for being present, patient and helpful, and also for his incredible and critical feedback and his empathic support during moments of uncertainty. His supervision and involvement contributed greatly to the quality and completion of my thesis.I am grateful for the opportunity to work with the optical Quantum Communications Lab team and for the creative freedom that I was granted to manage my own project.

As for my acadamic supervisor, Professor **Monia NAJAR**, There is a lot to be thankful for : her feedback, kind support, and motivation helped immensely, but topping off my list is for the incredible opportunity offered to carry out my internship in Portugal, at the University of Aveiro. What a wonderful experience and adventure ! I've acquired real world application and experience of my degree and I've met some amazing people. This has definitely been an experience I will never forget and always carry with me. I will forever be thankful.

I would like to express my sincere gratitude to **Zeinab RAHMANI** a PhD Student who is always encouraging, guiding, and assisting me whenever and wherever I end up facing difficulties. She has always made time to discuss my research and has offered suggestions and hints to assist me in overcoming any obstacles I have encountered.I have learned so much from you and I appreciate the time you spent teaching me new skills and useful thoughts. It was a pleasure to work with you because I was able to learn so much.

I also express my deepest thanks to **Manuel B. SANTOS**,a PhD Student for giving necessary advices and guidance while I was implementing the QOK. I choose this moment to acknowledge his contribution gratefully.

Special thanks go to my friend and my lab partner, **Yulianna GONCHAROVA**, for making my journey more enjoyable. Thank you for the terrific four months.

Last but not least, I would like to thank my family : my father, Mr. Habib Ben Khalifa ; my lovely mother, Mme. Saoussen Baccour ; my dear brothers, Houssem and Oussama for supporting me spiritually throughout writing this thesis and my life. I can't thank you enough for encouraging me throughout this experience.

Oumayma BEN KHALIFA

Table des matières

List of Figures

List of Tables

Nomenclature

— **2PC** = 2-party computation

— **AES-NI** = Advanced Encryption Standard New Instructions

— **AU** = Application Unit

— **BM OT** = Bellare-Micali OT

— **Bin .SS** = Binary Secret Sharing

— **DH** = DiffieHellman

— **DSRC** = Dedicated Short Range Communication

— **FCC** = Federal Communication Commission

— **GC** = Garbled Circuits

— **GCT** = Garbled Computation Table

— **GF** = Galois Fields

— **GPS** = Global Positioning System

— **HE** = Homomorphic Encryption

— **HSS** = Homomorphic Secret-Sharing

— **I2I** = Infrastructure-to-Infrastructure

— **IT** = Instituto de Telecomunicações

— **ITSs** = Intelligent Transportation Systems

— **LAN** = Local Area Network

— **LSSS** = Linear Secret-Sharing Scheme

— **MANETs** = Mobile Ad Hoc Networks

— **MP-SPDZ** = Multi-Protocol SPDZ

— **NPCs** = non-player characters

— **OBU** = On Board Unit

— **OSI** = Open Systems Interconnection

— **OT** = Oblivious Transfer

— **PDA** = Personal Digital Assistan

— **PoI** = Points Of Interest

— **QOKD** = Quantum Oblivious Key Distribution

— **QOT** = Quantum Oblivious Transfer Protocol

— **QROT** = Quantum Random Oblivious Transfer Protocol

— **QUESTS** = Quantum Enabled Security And Privacy In Vehicular Networks

— **RSU** = Road Side Unit

— **SCH** = Service Channel

— **SMC** = Secure Multi-party Computation

— **SPDZ** = Damgård, Pastro, Smart, and Zakarias

— **SS** = Secret Sharing

— **V2I** = Vehicle-to-Infrastructure

— **V2V** = Vehicle-to-Vehicle

— **VANETs** = Vehicular Ad Hoc Networks

— **WAN** = Wide Area Network

General Introduction

In recent years, the new development and enhancement in intelligent transportation systems (ITSs) have gained significant attention from both industry and research communities. The ITS contributes the major part in terms of providing road safety, improving traffic flow, and offering entertainment services on vehicles.

The automotive industry realizes the need for the vehicles to be connected with the wireless communication system, which enables communication between vehicles and between vehicles and infrastructure. Such communication can significantly increase traffic safety and optimize the traffic flow. Therefore, embedded sensors have been introduced in which traffic information such as driving behaviors, traffic flow parameters, and driving conditions can be shared with nearby vehicles by developing networks, which are named Vehicular ad hoc networks (VANETs).

Apart from assisting drivers to drive safely, VANETs can also provide infotainment to drivers for a more enjoyable driving as well as riding experience by offering comfort and commercial services.

It is in this context that our thesis project was carried out from March to July 2022 at the It-Instituto de Telecomunicações of the University of Aveiro in Portugal, with the aim of showing an overview of a quantum SMC use case implementation for VANETs and a discussion on its security and operational concerns. This Thesis is structured mostly into four main chapters dealing with the above mentioned work-plan – and is presented as follows:

— Chapter 1 gives an overview of the background to this project by introducing the host university, outlining the problem we are trying to solve and the proposed solution.

— Chapter 2 presents the state of the art of different aspects of VANETs, the concept of secure multi-party computation, as well as the exploration of a recommendation service based on location privacy protection in VANETs, and a discussion on its security and operational issues.

— Chapter 3 begins by discussing the context of the system to be developed, then a specification of the functional and non-functional requirements then ends with conceptual Model in which we present some graphical presentation of a set of elements of our system.

— Chapter 4 describes the implementaion work of the proposed solution,the working environment and the technologies used.

Finally, we conclude this work by discussing its contributions, limitations, and offering suggestions for some future work.

Chapter 1

General Context

Contents

Introduction

In this chapter, the project will be placed in its general context. We will start by presenting the environment of the training period by introducing the host institution that adopted this final year project. Then, we will present a description of the topic to be dealt with and the methodology used to solve the problems of this work.

1.1 Project context

This project is about implementing a quantum SMC use case for VANETs. It is done as part of the preparation of a graduation project for an engineering degree in the development of Communication Infrastructures and Services at the Higher Institute of Computer Science in Tunisia (ISI Ariana) for the academic year 2021/2022.

1.2 Host Institution

Instituto de Telecomunicações (IT)[1] is a private, not-for-profit organization, of public interest, a partnership of nine institutions with research and development in the field of Telecommunications.

— IT is actively involved in fundamental and applied research both at national and international levels,

— IT also plays its role towards public society with public awareness initiatives, knowledge transfer to industry, and by providing consulting services on a non-competing basis.

1.2.1 Mission

IT's mission[2] is to create and disseminate scientific knowledge in the field of telecommunications, which presupposes the development of fundamental and applied research activities in an international context, as a means to raise the level of education and training, both graduate and postgraduate, and to increase the competitiveness of Portuguese industry and telecommunications operators.

1.2.2 Activity

IT activities are centered on three poles, in Aveiro, on the University of Aveiro Campus, in Coimbra, on Pole II of the University of Coimbra and in Lisbon, at IST, as well as an external laboratory at the University of Covilhã and a delegation in Leiria at the Polytechnic Institute.

The main research areas are:

— wireless communications

— Optical Communications

— Networks and Multimedia

1.2.3 Organizational Chart

— IT is managed by a Board of Directors, with the support of the General Director , and by site management boards as shown in Figure 1.1[3].

— IT scientific activities are overseen by the Scientific Board, which meets twice a year to comment on the Work Plan and Budget, on the Annual Report, and on any other subject put forward by the Board of Directors.

— IT Work Plan and the Annual Report are discussed with the Advisory Council.

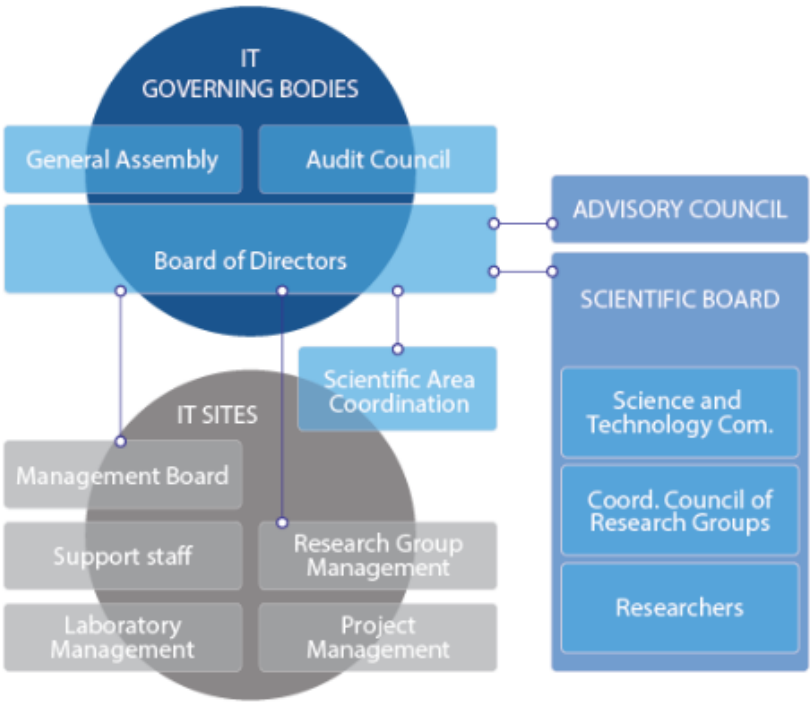


Figure 1.1: Organizational Diagram of Instituto de Telecomunicações

→ As for our project, it is part of the QUESTS[4] project which is an internal project from Instituto de Telecomunicações (IT) is going to develop a new generation of cryptographic technologies, based on the realization of both high-speed and secure quantum oblivious transfer (OT) cryptographic primitive. And regarding our training it was held in the Optical Quantum Communications Group with the QUESTS researchers team members.

1.3 Problematic

Over the past decade, there has been a growing interest in Vehicular Ad Hoc Networks (VANETs). These technological progresses allow the integration of VANETs not only to provide a safer and more enjoyable driving experience, but also to provide many other useful services to the driver and passengers of a vehicle.

With the popularisation and promotion of VANETs and positioning technology in everyday life, positioning accuracy is getting higher and higher, and location-based services are becoming more and more abundant. Some problems, such as information security and privacy leakage, have also emerged. Vehicles have become another way for attackers to steal user privacy.

Location-based services offer a variety of services to vehicles, but they can also bring security risks which will be discussed in the next subsection through a critical analysis of the most obvious security attacks and other techniques that threaten location privacy in Vanets.

1.4 Study of the existing system

It is important to understand the concept of privacy and security in a vehicular network. To this end, the following are the security threats and privacy requirements, we also discuss in particular the importance of location information and its privacy challenges in VANETs.

1.4.1 Type of attackers in VANETs

The role of the attacker is important in the vehicular network because of the launching of different types of attacks. The aim of the attackers is to create problems for other users of the network by changing the type of content of the messages. The attackers can be classified as follows according to the scope, nature and behaviour of the attacks [5]:

Passive Attackers These attackers only listen to the wireless channel to collect traffic information that can be passed on to other attackers.

As these attackers do not participate in the network communication process, they are called passive attackers.

Active Attackers These attackers generate packets containing wrong information or do not transmit the incoming packets.

Insider Attackers These attackers are authentic network users and have detailed knowledge of the

network. When they have all the information about the configuration, it is easy for them to launch attacks and create more problems than other attackers.

Outsider Attackers The outsider attacker is considered to be a authentic user of the network. This is a type of intruder that aims to misuse the network protocols and the scope of these attacks is limited. These attackers create fewer problems than insider attackers.

Malicious Attackers These attackers do not personally benefit from the attack. Their goal is to harm other network members or disrupt the functionality of a VANET. These attackers are considered the most dangerous category because they can cause serious damage to the network.

1.4.2 Attacks and security Threats in VANETs

With the emergence of VANETs, vehicles and drivers are now exposed to cyber security threats, compromising network access, anonymity, data stability and privacy, as well as the physical security of both passengers and facilities. We have reviewed and classified the attacks on VANETs according to their characteristics to make the following diagram: Figure 1.2.

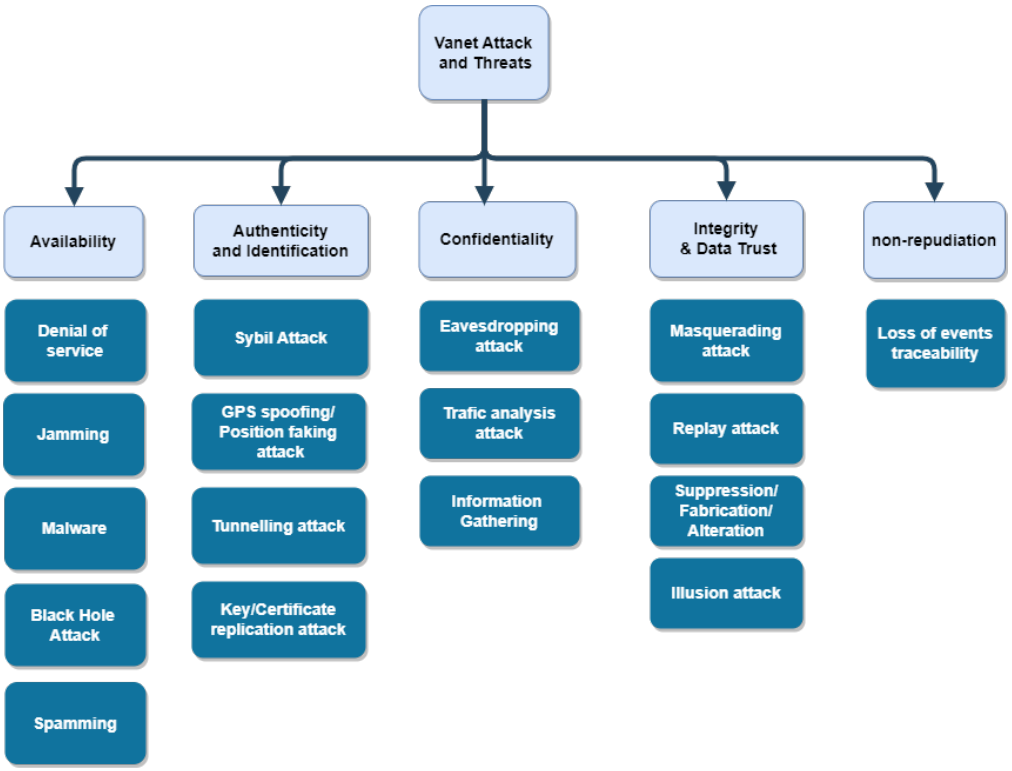


Figure 1.2: Vanet Attacks and Threats.

Attacks on availability Availability is the most important factor for VANETs. It ensures that the network is functional and useful information is always available during functioning time. This critical security requirement for VANETs, whose main purpose is to ensure the life of users, is a major target for most attackers. Several attacks fall into this category, the most famous are Denial of Service (DoS) attacks.

Attacks on authenticity and identification Ensuring authenticity in a vehicular network is a major challenge in VANET security to protect authentic nodes from external or internal attackers who infiltrate the network using a falsified identity. The importance of the identification and authentication process stems from the fact that it is frequently used when a vehicle needs to join the network or a service. There are several types of attacks in this category.

Attacks on confidentiality Confidentiality is an important security requirement for VANET communications, it ensures that data is only read by authorised parties. Without a mechanism to ensure the confidentiality of data exchanged between nodes in a VANET, the messages exchanged are particularly vulnerable to attacks such as the misuse of clear information.

Attacks on integrity and data trust The integrity of data exchanged in a system is to ensure that the data has not been changed in transit. Integrity mechanisms are therefore used to protect information from modification, suppression or insertion attacks. In the case of VANETs, this category mainly targets V2V communications as opposed to V2I communications because of their fragility. One possible technique that facilitates this type of attack is the manipulation of in-Vehicule sensors.

Attacks on non-repudiation In security terms, non-repudiation means the ability to verify that the sender and receiver are the entities claiming to have sent or received the message respectively.In a VANET context and given that the data handled is related to the security and privacy of users, it should always be possible to check all changes in security settings and applications (update, modification, addition, etc.) made to software and hardware.

1.4.3 Importance of Location Information

Location information [6] is one of the most crucial aspect of a vehicle and its driver. Since the VANET collects location information, this information must be kept confidential, otherwise attackers can gain quick access to the driver and compromise his or her privacy, allowing attackers to quickly attack the VANET and disrupt network efficiency.

In a secured VANET, data must be securely exchanged for it to function properly, and messages must be transferred between authorised parties only. However, if the attacker targets attacks such as eavesdropping, then the VANET data can be compromised, and the attacker can easily access location information.

1.4.4 Location Privacy Challenges in VANET

Privacy intrusions are related to unauthorised access to sensitive information in the vehicle. There is a direct relationship between the driver and the vehicle. If intruders gain unauthorised access to certain data, the driver's privacy will be compromised. In most cases, the owner of the vehicle is also its driver; if an attacker obtains the identity of the owner, the privacy of the vehicle may be endangered; this form of privacy invasion is well-known as identity disclosure. One of the most famous threats to privacy is location tracking. In this attack, the position of the vehicle or the path taken by the car at a particular time is considered personal data.

→In VANETs, protection must ensure that the communication traffic exchanged is not intercepted or manipulated by attackers.

1.5 Proposed Solution and Overall Project Objectives

In real life, when a driver arrives at an unfamiliar place, he or she really wants to have reliable service to recommend some potential places of interest nearby such as parking lots, gas stations, shopping malls, hotels, fast-food restaurants, etc. What all these applications have in common is that they are all location-based.

In order to overcome the observed shortcomings related to location privacy, we proposed to build and implement a new solution for VANETs drivers that offers the ability to identify different points of interest (PoI) nearby drivers based on the concept of Secure Multiparty Computation (SMC) and Quantum technology to make the solution faster and secure even against a quantum computer attack.

1.6 Task Planning

To plan our project properly and to facilitate the progress follow-up, we proceeded to build a work schedule by using a standardised representation technique.

→ To do this, we used the Gantt chart shown in **Appendix 1 - Gantt Chart**, which provides a visual view of the project's scheduled tasks, activities,and resources. It is a useful way of showing the work planned to be done on specific days.

Conclusion

This chapter has been devoted to present the host institute: Instituto de Telecomunicações (IT) and the context of this end of study project, to analyse the existing system in order to identify its shortcomings and to propose the eventual solution. At this point, we will present the VANETs architecture and also define the different concepts related to our solution, which will be the main object of the second chapter.

Chapter 2

State of The Art

Contents

Introduction

This chapter presents the theoretical aspects of this thesis. It provides the necessary background on the basic concepts of VANETs, its architecture, its characteristics and more precisely the related security concepts and methods. Furthermore, it presents the main analyses and approaches adopted to introduce the concept of secure multi-party computing and the associated protocols.Finally, we will outline the use of the Quantum technology.

2.1 Literature review on VANET architecture and applications

Traffic accidents [5], traffic congestion, fuel consumption and environmental pollution due to the large numbers of vehicles have become serious global problems. Traffic incidents are persistent issues in both developed and developing countries, resulting in huge loss of human lives and property.To meet these needs and make the journey safer, more efficient, hassle-free and more enjoyable, Intelligent Transport Systems (ITS) have introduced VANETs to create a safer infrastructure for road transport.

Lately, with the advancement of technology, more and more vehicles are equipped with GPS and Wi-Fi devices that are connected in a self-organising way, allowing vehicle-to-vehicle (V2V) communication, forming a vehicle ad hoc NET work (VANET).

VANETs are a subset of mobile ad hoc networks (MANETs) in which the communication nodes are mainly vehicles. With this network, one has to deal with a large number of highly mobile nodes, eventually spreading over different roads. → Simply put, VANETs focus primarily on road safety and efficient traffic management on public roads, while providing comfort and entertainment to drivers and passengers throughout their journeys.

2.1.1 Vehicular communications

As shown in figure 2.1 vehicular communication in VANETs can be achieved by exchanging information using:

— **Inter-vehicle communication** : This method is also known as vehicle-to-vehicle (V2V) communication or pure ad hoc network. In this method, vehicles communicate to each other without infrastructure medium. Information gathered by sensors in one vehicle or communicated to one vehicle can be directed to adjacent vehicles,

— **Vehicle-to-roadside communication** : This type of communication is also called vehicle to

infrastructure (V2I) communication. In this category, vehicles can use cellular gateways and wireless LAN access points to connect to the Internet and enable in-vehicle applications,

— **Inter-roadside communication** : This is also known as Infrastructure-to-Infrastructure communication (I2I). Vehicles may use infrastructure to communicate with each other and exchange information received from the infrastructure or from other vehicles through ad hoc communication.

Furthermore, vehicles can communicate with the infrastructure in single or multi-hop mode, according to their location, whether they are moving or not. This architecture includes V2V communication and offers more flexibility in terms of content sharing and enhances the network's reliability.

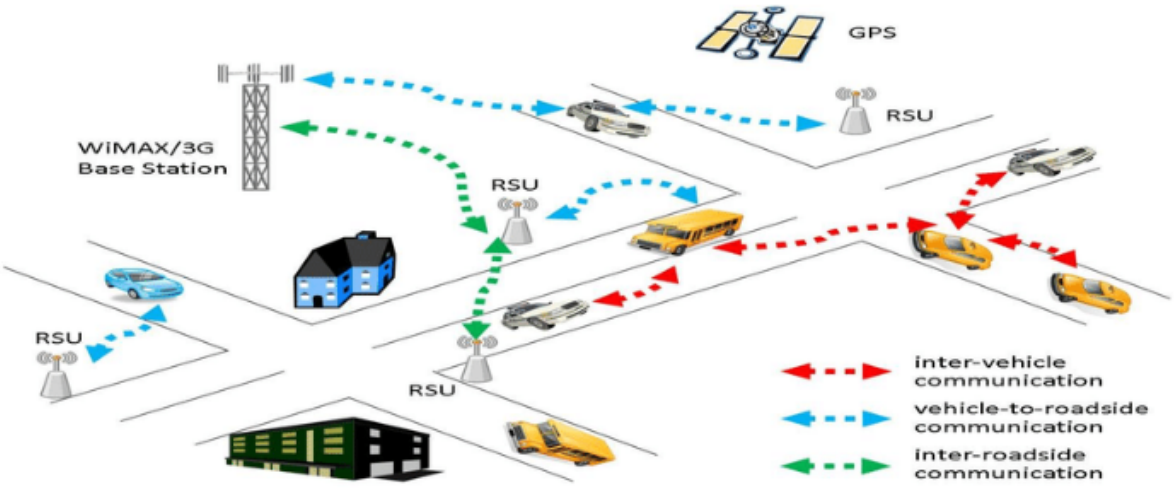


Figure 2.1: VANETs communication between V2V, V2I and I2I

2.1.2 VANET Standards

VANET standardization [7] affects all layers of the open system interconnection (OSI) model, which is used as a communication tool and includes all necessary features of all the layers. The dedicated short range (DSRC) communication, WAVE, and IEEE 802.11p are used to designate the full standard of communication protocol to deal with VANETs.

— Dedicated Short Range Communication (DSRC): DSRC is a wireless communication technology tool that permits vehicles to communicate with each other in ITS or other infrastructure such as V2V and V2I. In 1999, the Federal communication commission (FCC) allocated the band from 5.850 to 5.925 GHz, with a spectrum of 75 MHz for DSRC . As shown in **Figure 2.2**,

the spectrum of 75 MHz DSRC is sectioned into seven channels, and starts from Ch 172 to Ch 184. The Ch 178 is the control channel that can support the safety power applications. The other six channels such as 172, 174, 176, 180, 182, and 184 are the service channel (SCH). The Ch 172 and Ch 184 are used for high power and public safety messages, while other channels can be used to send both as a safety and non-safety messages.

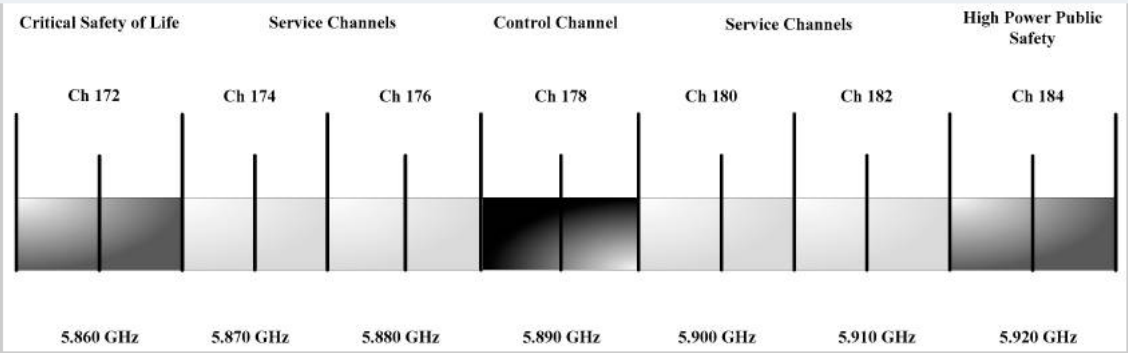


Figure 2.2: Channel diagram of dedicated short range communication (DSRC)

— WAVE: According to the IEEE published materials for the latest ITS standards, the WAVE IEEE 1609 describes an architecture, mechanism, sets of protocols, and interface, which are used to develop communication in the VANET environment, i.e., V2V and V2I communications. **Figure 2.3** shows the different standards of the WAVE architecture and its integration with the open system interconnection (OSI) model.

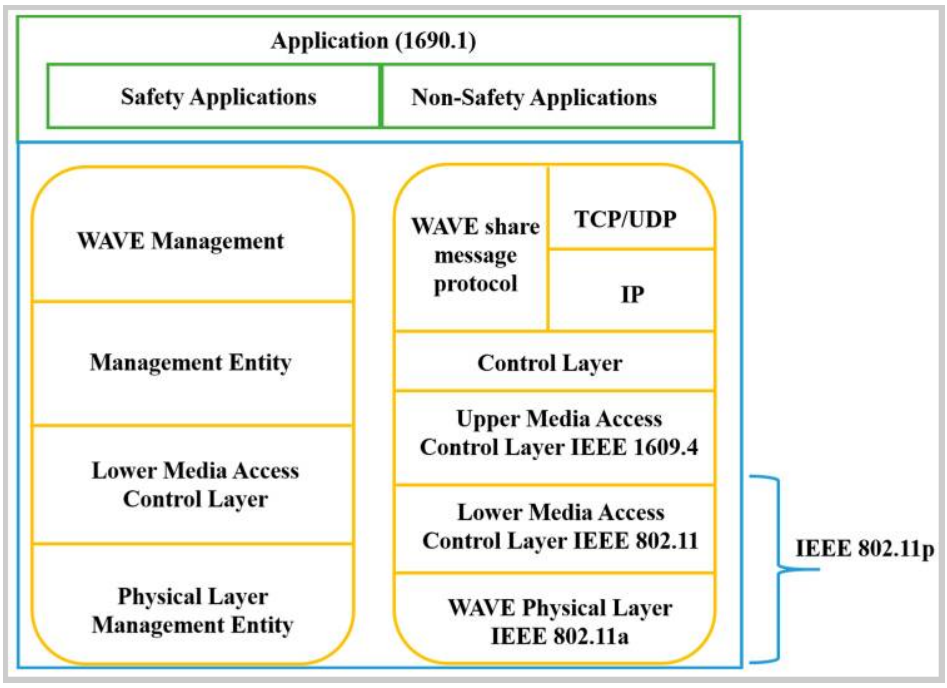


Figure 2.3: Wireless access in vehicular environments (WAVE) architecture

— IEEE 802.11p: After introducing the IEEE 1609 standards, the IEEE extended the family of IEEE 802.11 protocols by adding a new member 802.11p, which is used to facilitate the vehicular communication network. This is in compliance with the dedicated short range communication (DSRC) band.

2.1.3 VANET Architecture

WAVE [8] is a means of wireless communication between vehicles and between a vehicle and a roadside unit (RSU). Apart from providing drivers and passengers with a wide range of information, this communication method allows safety applications to enhance road safety and provide a comfortable driving experience.

The user and the provider are terms used to describe two different entities. **provider** provides the services while a **user** uses them.Depending on their role in the network, RSUs and OBUs may act as providers or users.

The main system components are the application unit (AU), (OBU) and (RSU) as depicted in **figure 2.4** [9] .

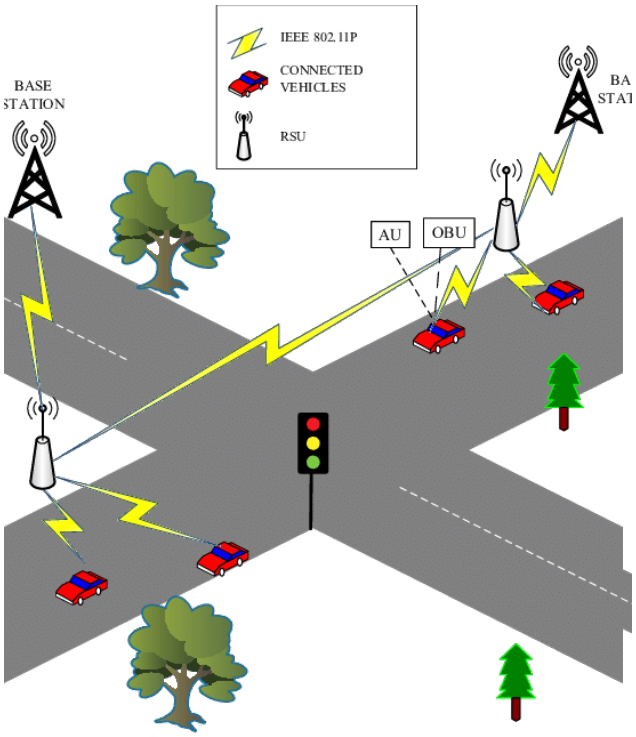


Figure 2.4: General Vanet System Architecture

On Board Unit (OBU): An On-Board Unit (OBU) is a piece of hardware installed in every vehicle used for exchanging information with RSUs or with other OBUs.

The main functions of the OBU are reliable message transfer, wireless radio access, ad-hoc and geographical routing, Network congestion control, data security and IP mobility.

Application Unit (AU) : An AU is a device installed inside the vehicle that is used with the application provided by the provider to communicate with the OBU. Apart from safety applications, the application unit can also be used by a normal device, e.g. a personal digital assistant (PDA) running online services. The OBU is connected to the application units via a wired or wireless medium and can be located with the OBU in a single physical unit; the distinction between the UA and the OBU is logical.

Road Side Unit (RSU) : The RSU is a wave device usually fixed along the roadside or in specific positions, such as near intersections and parking lots. The RSU is equipped with one network device for a dedicated short range communication based on IEEE802.11p radio technology, and can also be equipped with other network devices so as to be used for the purpose of communication within the infrastructural network and can be used also to prevent accidents. The information can only be accessed by an authenticated user.We use the techniques of pseudonyms, mix zones, ad hoc anonymity and silent periods. For example, they are located near intersections and car parks, where there is a high density of vehicles.

2.1.4 VANET Applications

In a VANET environment, drivers and travellers can access a wide range of information and develop a large number of applications. These applications are classified by their main purpose: Applications that increase safety of vehicles on the road are called safety applications. Applications that provide valuable services, such as entertainment, are called user applications.

2.1.4.1 User applications

Passengers in vehicles that spend a very long time in transit may be interested in some of the application areas for vehicle networks to provide many different types of information. These can provide vehicle users with a variety of information, announcements and entertainment during their journey.

The main purpose of such applications is to improve passenger comfort and traffic efficiency.They can provide drivers or passengers with weather and traffic information, and identify various points of interest (PoIs) such as car parks, petrol stations, shopping centres, hotels, fast food outlets, etc.

The use of VANET for commercial and comfort applications is considered detrimental to traffic safety and efficiency. It also distracts and interferes with safety-related applications.

2.1.4.2 Safety applications

The main goal of the safety applications is to increase public safety and protect the loss of life by providing time-sensitive, lifesaving traffic information to drivers,these applications use wireless communication between vehicles or between vehicles and infrastructure to improve road safety and avoid crashes with other vehicles on the road, with the intention of saving people's lives and providing a clean environment.

It is a key requirement of these applications that the safety data must be transmitted to the intended receivers (vehicles approaching the dangerous zone) within a limited time.

Driver assistance, alert information, and warning alerts are the three basic VANET safety applications.

→ As for our project, we are more likely in the first category which is the user application because it is an application that helps the driver to identify different points of interest according to his needs.

2.1.5 VANET Characteristics

VANETs [8] are quite different from other networks in their behaviour and characteristics.

High mobility VANET operates in a very dynamic environment. Traffic density is higher during rush hours, with vehicles moving at different speeds.

The main elements of a VANET are the fixed RSUs and the mobile vehicles. Due to the variable speed of the vehicle, communication issues arise at very low and very high speeds. Indeed, heavy traffic jams slow down or stop the traffic, so that vehicles have time to exchange messages.

Predictable mobility Roadways are the only roads where vehicle movements can be predicted. With GPS, it is easy to find information about roads. The position of a vehicle can easily be determined by watching its speed and its trajectory on the road.

Rapid Topology Changes Due to the dynamic movement of vehicles, the topology of the VANET changes rapidly.

Node heterogeneity There are a wide range of different types of vehicles, such as emergency

vehicles, authority vehicles transport vehicles, and more.

Node density The number of nodes in the network can vary greatly. Sometimes the roads may be completely empty of vehicles, while at other times every space on the road may be occupied. A high density of nodes means more messages to send and process.

Network connectivity The topology is constantly changing as vehicles change position.

This is the direct result of high mobility. Connectivity between nodes is short and changes frequently. Nodes can disconnect at any time. High packet loss must be avoided.

Battery Power and Storage Capacity The power restraint is no longer an issue in VANETs, as vehicles have powerful batteries that provide constant power to the OBUs.

A network is constructed to allow the transmission of information from one vehicle to another without the need for energy or computing resources. They also have their own batteries and extremely powerful computers to handle complex calculations.

→ So far, we have given a general overview of VANET architecture, its types of communication, its types of applications and its main characteristics. But we have also noticed in the study of the existing situation in Chapter 1 that VANET poses many security and privacy issues and the automotive industry is very reluctant to use them. Multi-Party Computation (MPC) is a cryptographic technique which allows a set of parties to compute the output of a function while not revealing the input. This next section investigates the potential and feasibility of multi-party computation to solve the present problems of autonomous vehicle communication.

2.2 Secure Multiparty Computation

Secure multiparty computation is a cryptographic service that allows untrusted parties to compute a function retaining the privacy of their inputs and outputs. The privacy of one party input and output, should only be limited by what is revealed by the other parties' outputs. Secure multiparty computation also restrict the usage of the parties' inputs to a specific computation. Let's assume that there are N entities that want to compute something, that can be described by the function $f()$, that depends on the input parameters, $x_1, x_2, \ldots, x_N$, which are distributed among N entities. This computation can be described by the following equation,

$$(y_1, y_2, \cdots, y_N) = f(x_1, x_2, \ldots, x_N) \tag{2.1}$$

The computation of function $f()$, it is in principle a simple task as long as the parties do not care about the privacy of theirs inputs and outputs. **figure 2.5** ,shows a pictorial representation of this computation. The computation can be executed by each entity, if they all share the inputs among them and if all are in possession of the function $f()$. Alternatively, the computation can be performed by an external entity that knows the function $f()$ and that collets the inputs from all the parties. In this case, the external entity performs the computation and shares the outputs.

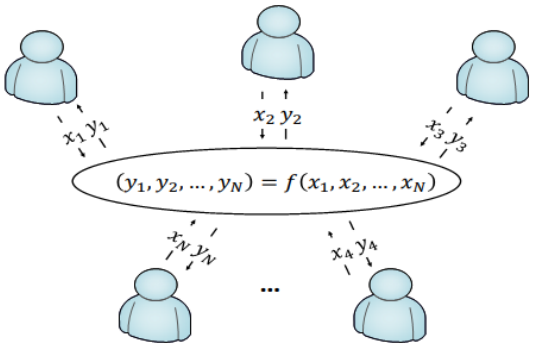


Figure 2.5: Computing a function that depends on data distributed between N parties.

However, if each entity wants to preserve the privacy of its own input/output pair and wants to make sure that its own data is only going to be used for that specific purpose, i.e. the computation of that specific function $f()$, that is a more challenging task. Nevertheless, this is still easily achieved as long as the N entities can agree on a trusted external entity. This external entity should collect all inputs, perform the computation, and distribute the inputs and outputs accordingly. At the end the inputs and outputs should be destroyed to preserve their privacy and to make sure that they cannot be reused in any future computation. In **figure 2.6** shows a pictorial representation of this computation, where the external trusted entity assumes a central role.

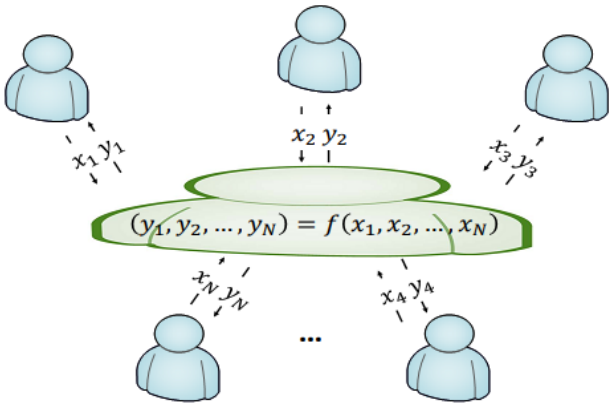


Figure 2.6: Computing a function using an external trusted entity to preserve privacy.

In the previous scheme, the trusted external entity collects the inputs, performs the computation, distributes the outputs, and destroys the inputs and outputs. The trusted external entity is a critical element of the system. If the external entity is corrupted, or if the N entities cannot agree in a entity to assume this role, the security is compromised or the computation cannot be performed. Notice also that frequently due to legal restrictions some entities are obliged to protect the privacy of the data in their possession by law, and by law they cannot share the data that are in their possession with external entities, trusted or not. The purpose of secure multiparty computation is to allow this cryptographic service, without the need of an external trusted entity but with the same level of privacy and correctness that an ideal trusted external entity would provide.

We can think of a generic secure multiparty computation service assuming a scenario where each entity is going to be empowered with an encoder/decoder device that is able to encrypt its input, x_j , and to decrypt its output, $\tilde{y_j}$. In this new scenario the input data is shared in an encrypted way, $\tilde{x_1}, \tilde{x_2}, \ldots, \tilde{x_N}$, in a way that the plain information cannot be accessed by any other entity, with exception of the one that encrypted it. The outputs are also encrypted, $\tilde{y_1}, \tilde{y_2}, \ldots, \tilde{y_N}$, in a way that make each plain output, y_j , only available for its destination party, j. Besides that, we still need a new function $\tilde{f}()$, able to compute $\tilde{y_1}, \tilde{y_2}, \ldots, \tilde{y_N}$, from the inputs $\tilde{x_1}, \tilde{x_2}, \ldots, \tilde{x_N}$. Mathematically, this encrypted computation can now be described by

$$(\tilde{y_1}, \tilde{y_2}, \cdots, \tilde{y_N}) = \tilde{f}(\tilde{x_1}, \tilde{x_2}, \ldots, \tilde{x_N}), \tag{2.2}$$

With

$$\tilde{x_1} = E_1(x_1), \cdots, \tilde{x_j} = E_j(x_j), \cdots, \tilde{x_N} = E_N(x_N), \tag{2.3}$$

and

$$y_1 = D_1(\tilde{y_1}), \cdots, y_j = D_j(\tilde{y_j}), \cdots, y_N = D_N(\tilde{y_N}) \tag{2.4}$$

where $\tilde{x_j}$ and $\tilde{y_j}$ are encrypted versions of x_j and y_j , respectively, related through the j encoder, $E_j()$, and decoder, $D_j()$, respectively. Notice, that by now, we were able to reduce the secure multiparty computation implementation problem to two sub-problems. The first one is to design the encoder/decoder able to obtain $\tilde{x_j}$ from x_j , and y_j from $\tilde{y_j}$, respectively. The second one is to design the function $\tilde{f}()$, that computes **2.2**. The problem that we face is how to obtain $\tilde{f}()$ from a generic function $f()$, and how to obtain $\tilde{x_j}$ from x_j , in a way that using all $\tilde{x_j}$ as inputs of $\tilde{f}()$ we will obtain a set of outputs $\tilde{y_j}$, from which each entity can obtain its and only its output

y_j . With this, the entities can share their encrypted inputs, $\tilde{x_j}$,among them, and all entities can be empowered with the encrypted function, $\tilde{f}()$, and each one can compute the encrypted outputs. Keep in mind that because each entity is only able to decrypt its own output, $\tilde{y_j}$, it does not have access to the others decrypted outputs, $y_l \neq j$.

Figure 2.7 shows a pictorial description of this cryptographic secure multiparty computation service, where each entity is empowered with an encoder/decoder and the function that computes the encrypted outputs from the encrypted inputs, $\tilde{f}()$, is assumed to exist.

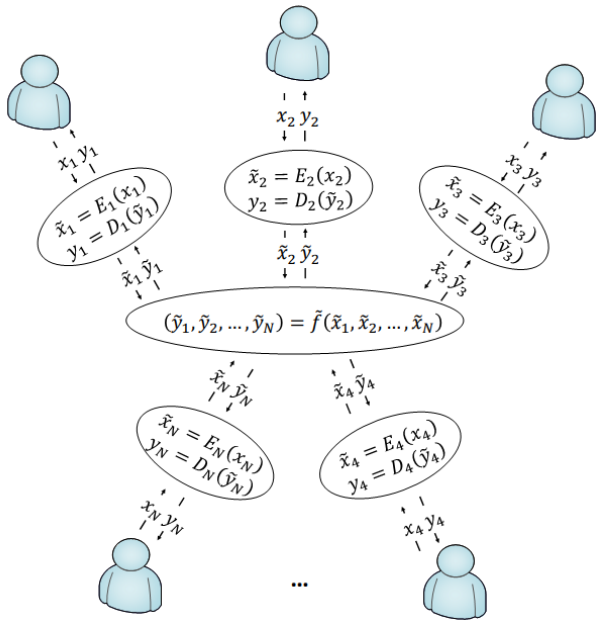


Figure 2.7: Computing a function while preserving the confidentiality of inputs and outputs by using an encoding/decoding device.

2.2.1 Illustrative MPC example

For example [10], let's say three people, John, Rob and Sam, want to know who has the highest salary without revealing to each of them how much they earn. – this is actually a classic example of MPC, known as The **Millionaire's Cryptography Problem** [11]. By simply using their own salaries $(d1, d2, d3)$, they want to find out which salary is the highest and not share any real numbers between them. Mathematically, this translates into a calculation of :

$$F(d1, d2, d3) = max(d1, d2, d3)$$

If there were a trusted third party (i.e. a mutual friend who they know can keep the secret), they could each communicate their salary to that friend and find out which of them earns more,

also known as $F(d1, d2, d3)$, without ever knowing the private information. MPC's goal is to design a protocol in which, by only exchanging messages with each other, John, Rob and Sam can always learn $F(d1, d2, d3)$ without revealing who earns what and without having to rely on an external third party. They should learn no more by engaging in MPC than they would have by interacting with their trusted mutual friend.

2.2.2 Types of parties in SMC

It is common in SMC to classify parties into three categories: **input parties**, **calculation parties** and **output parties**. Input parties provide data for the computation. The calculation parties perform the calculation while preserving the confidentiality of the data. Apart from what is revealed by the architecture of the calculation, these parties must not know anything about the input data. Finally, there are the result parties who receive the results of the computation.

→ Parties can play several roles. In our research, autonomous vehicles play both the role of input and result parties.

2.2.3 Properties of secure multi-party computation

For MPC [12] to be safe and secure, some properties must be respected.The ideal/actual simulation paradigm is an efficient method to find these properties. The paradigm defines an ideal world where an independent party receives the private inputs, computes the functions and returns the result to everyone. This party is assumed to be absolutely honest and trustworthy. The scenario in the ideal world is safe by definition. By comparing the calculations in the ideal world with the calculations in reality, the paradigm shows that MPC must respect the following properties:

— **Privacy**: The only known input of a party should be its own. The only information available about the inputs of other parties is that can be derived from the output itself.

— **Independence of Inputs**: Honest and corrupt parties must choose their inputs independently of each other.

— **Correctness**: It is guaranteed that from the inputs a correct output will be calculated. The output will only be incorrect if the inputs are incorrect. An adversarial party must not be able to prevent other parties from receiving their correct outputs.

In this context, an **adversary** refers to the parties attacking the computational process. The purpose of the attack may be to learn private information from other parties or to make the

result of the calculation incorrect.

— **Guaranteed output delivery**: Corrupted parties should not be able to prevent honest parties from receiving their outputs. In other words, the adversary should not be able to interfere with the calculation by performing a denial of service attack.

— **Fairness**: The corrupted parties should receive their outputs if and only if the honest parties also receive their outputs. The scenario where a corrupted party gets an output and an honest party does not should not be allowed to occur.

2.2.4 Adversarial Models

A very important issue we need to pay attention to is the power of the adversary [13] attacking the execution of a protocol. As we have mentioned, the adversary controls a subset of the participating parties in the protocol. We describe the two main parameters defining the adversary's power: its behaviour as an authorised adversary, that is whether it simply collects information passively or whether it can instruct the corrupted parties to act maliciously, and its corruption strategy, that is when and how the parties come under the adversary's control.

2.2.4.1 Adversarial behavior

— **Semi-honest adversaries**: In the semi-honest adversarial model, even the corrupted parties follow the protocol specification correctly. However, the adversary obtains the internal state of all corrupted parties and tries to use it to learn information that should remain private. This is a quite weak adversarial model, but a protocol with this level of security ensures that there is no unintentional data leakage. In some cases this is sufficient, but in today's adversarial environment it is often not enough.

Semi-honest adversaries are also called "honest-but-curious" and "passive."

— **Malicious adversaries**: In this adversarial model, corrupted parties can deviate arbitrarily from the protocol specification according to the adversary's instructions. In general, it is preferable to ensure security in the presence of malicious adversaries, as this ensures that no adversary attack can succeed. Malicious adversaries are also called "active".

— **Covert adversaries**: Such an adversary may behave maliciously in an attempt to break the protocol. However, the security guarantee provided is that if it attempts such an attack, it will be detected with some specified probability that can be adjusted to the application. We would

point out that, unlike the malicious model, if the adversary is not detected, it may succeed in cheating (for example, learning the input of an honest game). This model is suitable for situations where a real penalty can be associated with the detection of an adversary, and where the adversary expects to lose overall if he attempts an attack.

2.2.4.2 Corruption Strategy

— **Static corruption model**: In this model, the set of parties controlled by the adversary is fixed before the protocol starts. Honest parties remain honest throughout the protocol and corrupted parties remain corrupt.

— **Adaptive corruption model**: Instead of having a fixed set of corrupted parts, adaptive adversaries have the ability to corrupt parts during computation. The choice of who to corrupt, and when, can be arbitrarily decided by the adversary and may depend on its view of the execution (it is called adaptive for this reason). This strategy models the threat of an external "hacker" breaking into a machine during an execution, or a party that is initially honest and then changes its behaviour. We note that in this model, once a party is corrupted, it remains corrupt from that point on.

— **Proactive security model**: This model considers the possibility that parties are only corrupted for a certain time. Thus, honest parties may be corrupted throughout the computation (as in the adaptive adversarial model), but corrupted parties may also become honest. The proactive model makes sense in cases where the threat is an external adversary that can penetrate networks and break into services and devices, and where secure computations are in progress. When breaches are discovered, systems are cleaned up and the adversary loses control of some machines, making the parties honest again. The security guarantee is that the adversary can only learn what it has derived from the local state of the machines it has corrupted, even though they have been corrupted. Such an adversary is sometimes called mobile.

2.3 Specific Functionalities of Interest

Here, we define several features that have been identified as particularly useful building blocks for the construction of MPC protocols.

2.3.1 Garbled Circuits

Yao's concept of garbled circuits is used in almost all MPC protocols that evaluate a Boolean circuit. The main idea is to let party A, called the **creator**, garble the function to be computed. The garbled circuit is sent to party B, called the **evaluator**. Then A gives B a key Xa that corresponds to A's input a. B requires the key Y_b associated to his input, but A should not learn what B's input bit b. This is done using the concept of Oblivious Transfer (OT) in which A inputs Y_0, Y_1, and B inputs b and learns Y_b, but A learns nothing. (This will be explained in more details in the next paragraph). If A wants to garble an AND gate, A computes four ciphertexts: $C_{00} = E_{X_0;Y_0}(Z0)$, $C_{01} = E_{X_0;Y_1}(Z0)$,$C_{10} = E_{X_1;Y_0}(Z0)$, $C_{11} = E_{X_1;Y_1}(Z1)$. They are permuted randomly and transmitted to B, who can only decrypt the ciphertext encrypted using the keys X_a, Y_b. B learns the output $Z_{a\wedge b}$ and the parties exchange it.

2.3.2 Oblivious Transfer

Oblivious transfer (OT) [14] is a cryptographic primitive defined as follows: in its simplest form, OT 1-out-of-2, a sender has two input messages M_0 and M_1 and a receiver has a choice bit c. At the end of the protocol, the receiver is supposed to learn message M_c and nothing else, while the sender is supposed to learn nothing. Perhaps surprisingly, this extremely simple primitive is sufficient to implement any cryptographic task. OT can also be used to implement most advanced cryptographic tasks, such as secure two-party and multiparty computation (for example, the millionaire's problem) in an efficient way.

The protocol is a simple modification of the well-known DiffieHellman (DH) key exchange protocol. Given a group G and a generator g, the DH protocol allows two players, Alice and Bob, to agree on a key as follows: Alice samples a random a, computes $A = g^a$ and sends A to Bob. Symetrically, Bob takes a random b, computes $B = g^b$ and sends B to Alice. Now both parties can calculate $g^{ab} = A^b = B^a$ from which they can derive a key k. The key observation now is that Alice can also derive a key different from the value $(B/A)^a = g^{ab-a^2}$, and Bob cannot compute this group element (assuming the DH computation problem is hard).

We can now turn this protocol into an OT protocol by letting Alice play the role of the sender and Bob the role of the receiver (with the choice bit c) as shown in **figure 2.8**. The first message (from Alice to Bob) remains unchanged (and can be reused on several instances of the protocol) but now Bob calculates B according to his choice bit c: if $c = 0$ Bob calculates $B = g^b$ and if $c = 1$ Bob

calculates $B = Ag^b$. At this point, Alice derives two keys k_0, k_1 from $(B)^a$ and $(B/A)^a$ respectively. It is easy to verify that Bob can derive the key k_c corresponding to his choice bit of A^b, but cannot compute the other. This can be considered as a random OT, this is an OT where the sender has no input but receives two random messages from the protocol, which can be used later to encrypt its inputs, thus realising the OT functionality.

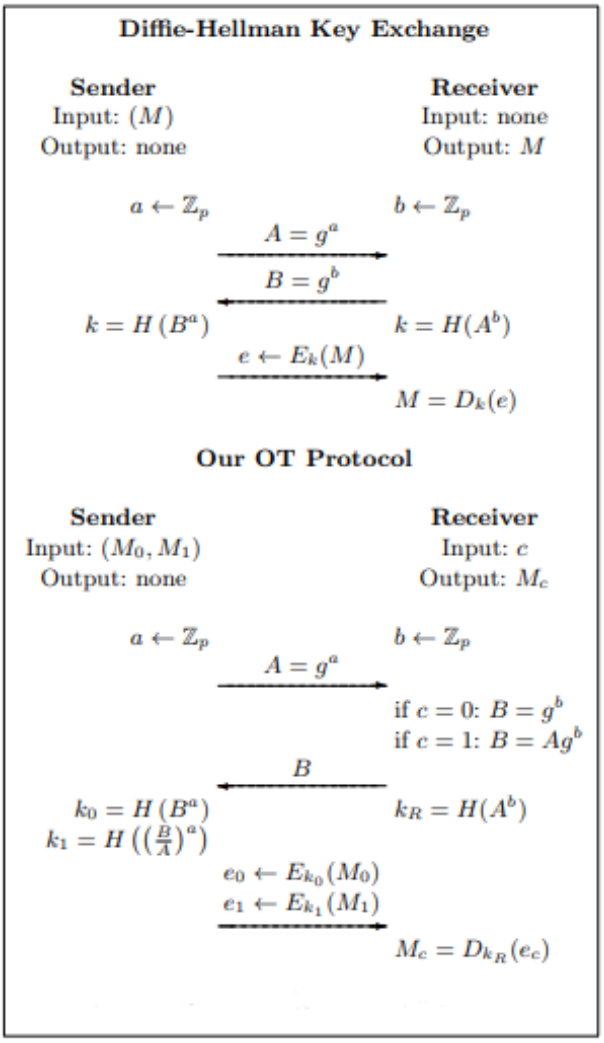


Figure 2.8: Oblivious Transfer

2.3.3 Homomorphic Encryption

The problem with encrypted data is that you have to decrypt it to use it. In doing so, it is vulnerable to the same things we tried to protect it from by encrypting it. There is a powerful solution to this scenario, which is homomorphic encryption [15].

Homomorphic encryption allows encrypted data to be analysed or manipulated without

revealing it to anyone by adding some algebraic manipulation, such as addition or multiplication that acts consistently between the plaintext and the ciphertext. In other words, if we multiply or add the ciphertext, the plaintext changes as well. At the same time, this relationship must be implemented in such a way as to be hidden from an observer. If watching mathematical operations on ciphertexts reveals information about the corresponding ciphertexts, then the encryption is broken.

It is very difficult to achieve these mutual goals of strong encryption and the ability to perform mathematical operations on ciphertexts and get the right answer. Homomorphic encryption algorithms are those that have achieved this goal.The difficulty is that designing such an encryption algorithm. Therefore, there are a few different types of homomorphic encryption that describe how close a particular algorithm is to this end goal, namely:

— **Partially Homomorphic Encryption**: keeps sensitive data secure by only allowing select mathematical functions to be performed on encrypted data ;

— **Somewhat Homomorphic Encryption**: supports limited operations that can be performed only a set number of times ;

— **Fully Homomorphic Encryption**: this is the gold standard of homomorphic encryption that keeps information secure and accessible.

2.3.4 Additive secret sharing

A secret sharing scheme [12] is a cryptographic tool that divides a secret into a number of shares, with one share being sent to each computional party. In an additive secret sharing scheme, the secret can only be reconstructed when all the shares are combined. This scheme is based on additive homomorphic encryption and can be formalized mathematically.

A secret s to be shared between n parties is divided in shares $(r_1, \ldots, r_{n-1}, r_n)$, where r_i is random for $i \in \{1, \ldots, n-1\}$, and $r_n = s - \sum_{i=1}^{n-1} r_i$. To recover the secret all shares have to be combined: $s = r_n + \sum_{i=1}^{n-1} r_i$.

2.4 What kind of MPC technology should We use?

A variety of secure multiparty computation (MPC) protocols have been proposed up to now as shown in **table** 2.1 [16]. Since their performance characteristics are incomparable, the most suitable MPC protocol may be completely different depending on the given computational task and environment. It is tedious work to compare all the possibility to choose the most suitable MPC.

Table 2.1: SMC Protocols.

Security Model	Mod prime/ GF(2^n)	Mod 2^k	Bin.SS	Garbling
Malicious,dishonest majority	MASCOT/LowGear/HighGear	SPDZ2k	Tiny/Tinier	BMR
Covert,dishonest majority	CowGear/ ChaiGear	N/A	N/A	N/A
Semi-honest,dishonest majority	Semi/Hemi/Temi/ Soho	Semi2k	SemiBin	Yao'sGC/ BMR
Malicious,honest majority	Shamir/Rep3 /PS/SY	Brain/ Rep3/PS/SY	Rep3/CCD/PS	BMR
Semi-honest,honest majority	Shamir/ATLAS/Rep3	Rep3	Rep3 / CCD	BMR
Malicious,honest supermajority	Rep4	Rep4	Rep4	N/A
Semi-honest,dealer	Dealer	Dealer	Dealer	N/A

— Modulo prime and modulo 2^k are the two parameters that allow an integer computation. For $k = 64$, the latter corresponds to the computation available on the widely used 64-bit processors.

— GF(2^n) denotes Galois extension fields with order 2^n, which are different from the modulo 2^n computation.

— Bin. SS stands for binary secret sharing, that is secret sharing modulo two.

— A security model specifies how many parties are "allowed" to misbehave and in which sense. Malicious means that non-compliance with the protocol will at least be detected, while semi-honest means that even corrupt parties are supposed to follow the protocol.

→ MPC protocols can often be classified into one of two classes: MPC protocols based on secret sharing and MPC protocols based on garbled circuits. Seeing all the progress made around MPC protocols, one may wonder how to choose an adequate secure multiparty computation among all this variety for a given purpose and environment. To accomplish this task, with a given circuit$C$ a set of parties P and an adversary A, there are a few things to consider in order to find the most efficient protocol [17] :

— How many parties are involved in the computations?

— What is the network type that the parties are connected to?

- If it is a fast network with a relatively low ping time (LAN), it is better to use secret sharing protocols (SS).

- If the parties are distant from each other and connected via a WAN, GC (Garbled Circuit) protocols are in most cases faster due to the lower complexity of the rounds.

— Some parties can be corrupted by A . What are the capabilities of these corrupt parties? This is where things can get a bit complicated, as we have to distinguish between semi-honest and malicious corruption:

- If there are two parties and one of them is maliciously corrupted, we need to use the SPDZ protocol (based on LSSS) or one of the protocols in the HSS/WRK line (based on GC) or dual execution.
- The case of the three-part honest majority has gained a lot of attention in the last 3-4 years. Libraries that support such features are ABY3, MP-SPDZ, SCALE-MAMBA and Sharemind.
- For more than three parties honest majority we can find protocols in MP-SPDZ and SCALE.

— Computational domain: **Arithmetic protocols** (modulo prime or power of two) are preferable for many applications as they allow for low cost addition and multiplication of integers. However, **binary circuits** may be a better option if there are very few integer computations.

— Secret sharing vs garbled circuits: Computing using secret sharing requires a number of communication rounds that increase with the computation, which is not the case for garbled circuits. However, the cost of computing integers as a binary circuit often compensates for this.

— Underlying technology for dishonest majority: While secret sharing alone is sufficient for the calculation of the honest majority, the dishonest majority requires either homomorphic encryption (HE) or oblivious transfer (OT). Both options offer a trade-off between computation and communication: while oblivious transfer is easier to compute, homomorphic encryption requires less communication. Moreover, the homomorphic encryption requires a certain number of batches to be efficient, which makes OT preferable for small tasks.

→ After considering our use case, taking into account all the points mentioned above, namely: the number of parts involved in our scheme (two-party), how many of them are corrupted (semi-honest adversaries), how they are connected to each other (ad-hoc network) and also considering the computational task we are trying to solve, we chose to work with Yao's protocol for 2-party computation (2PC) and we will describe its basic concept and advantages in this next section.

2.5 Yao's Garbled Circuits Protocol

This section provides a complete description of Yao's garbled circuits protocol and how the protocol incorporates OT.

2.5.1 Yao's Protocol Description

Yao's Garbled Circuits presented the first protocol to introduce secure (two-party) computation. It is a cryptographic scheme that allows two parties with private inputs x and y to jointly and securely compute any function $f(x, y)$ by transforming it into a garbled Boolean circuit $C(x, y)$. Considering that a garbled Boolean circuit is a collection of garbled Boolean gates.

The key elemnents of this protocol include :

— Constant number of rounds ;

— Secure only for semi-honest adversaries ;

— Many applications of the methodology beyond secure computation ;

— Privacy : Nothing is learned from the protocol other than the output ;

— Based on a Boolean circuit.

2.5.2 Yao's GC Construction

We now present a high-level description of Yao's protocol. The construction is actually a "compiler" that takes any polynomial-time functionality f, or actually a circuit C that computes f, and constructs a protocol for securely computing f in the presence of semi-honest adversaries. In a secure protocol, the only value learned by a party should be its output. Therefore, the values that are allocated to all wires that are not circuit-output, should not be learned by either party (these values may reveal information about the other party's input that could not be otherwise learned from the output). The basic idea behind Yao's protocol is to provide a method of computing a circuit so that values obtained on all wires other than circuit-output wires are never revealed. For every wire in the circuit, two random values are specified such that one value represents 0 and the other represents 1 [18].

For a better understanding, we will consider the example of an **AND Gate** [19] to understand first what is a single garbled gate, then we can easily generalize to the whole construction of a garbled circuit.

an AND Gate (Basic) ,

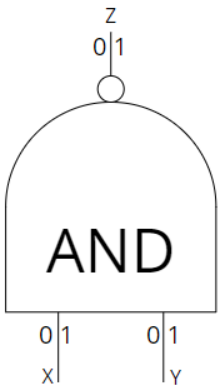


Figure 2.9: AND Gate

Table 2.2: AND Truth Table.

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

— x,y are called INPUT wires, z is called OUTPUT wire as shown in **figure 2.9** .

— For each of these three wires we have two values: 0, 1 as shown in the Truth **Table 2.2** (input values for the input wires, output values for the output wire).

— Once input values a; b are selected, we want to compute $g(a, b)$ securely.

↪ **an AND Gate with Garbled Values** ,

For each wire x, y, z, we specify two random values, corresponding to 0 and 1 as shown in figure 2.10 :

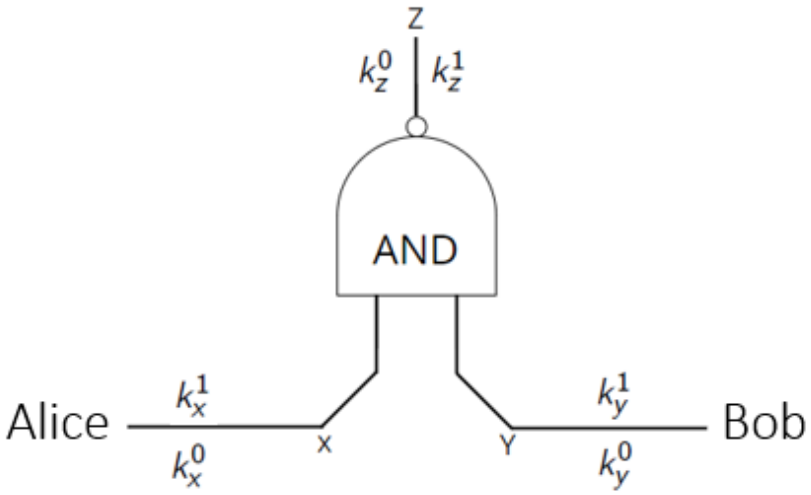


Figure 2.10: AND Gate with garbled values.

after we neeed to associate k_z^1, k_z^0 with k_y^0 ,k_y^1 ,k_x^0, k_x^1 using the Garbled Computation Table (GCT) shown in Table 2.3.

View k_y^0 ,k_y^1 ,k_x^0, k_x^1,k_z^1, k_z^0 as encryption keys and encrypt k_z^1, k_z^0 under appropriate pairs of input keys.

Table 2.3: Garbled Computation Table (GCT).

input wire x	input wire y	output wire z	GCT
k_x^0	k_y^0	k_z^0	$E_{k_x^0}(E_{k_y^0}(k_z^0))$
k_x^0	k_y^1	k_z^0	$E_{k_x^0}(E_{k_y^1}(k_z^0))$
k_x^1	k_y^0	k_z^0	$E_{k_x^1}(E_{k_y^0}(k_z^0))$
k_x^1	k_y^1	k_z^1	$E_{k_x^1}(E_{k_y^1}(k_z^1))$

Note : With given two input keys k_x^a and k_y^b , only one row of the GCT can be decrypted correctly, namely: $E_{k_x^a}(E_{k_y^b}(k_z^{g(a,b)}))$.

↪ **A Garbled AND Gate** ,

— Alice chooses two random keys for each wire, giving her 6 keys in total.

— She encrypts each row of the table, creating the GCT:

— She permutes it (rearranges it), so that the position of the key reveals nothing about the value to which it is associated.

$$\updownarrow \begin{cases} E_{k_x^0}(E_{k_y^0}(k_z^0)) \\ E_{k_x^0}(E_{k_y^1}(k_z^0)) \\ E_{k_x^1}(E_{k_y^0}(k_z^0)) \\ E_{k_x^1}(E_{k_y^1}(k_z^1)) \end{cases}$$

— She sends it over to Bob, along with her input key $k_x^{b'}$, with b' her input value.

— Bob still needs his own key to decrypt the GTC, so Alice must send it to him.

— If Alice sends both k_y^0; k_y^1 to Bob, then Bob can decrypt more information and if Bob asks Alice for the key that corresponds to her input, then Alice learns Bob's input.

The question now is : **how does Bob receive his key ?** bearing in mind that :

— Bob can't generate them by himself ,

— Bob can't ask Alice to provide them ,

— Alice can't give Both values for each input wire .

A solution to this problem is proposed in **Oblivious Transfer**, A protocol in which:

— sender inputs x_0, x_1 , receiver inputs s ;

— receiver obtains x_s ;

— sender learns nothing about s, receiver learns nothing about x_{s-1} .

↪ **Computation of the garbled gate** ,

— Bob now has $k_x^{b'}$; k_y^b and the GCT and he can compute the gate by decrypting GCT.

— He can decrypt only one line of the GCT, exactly because of its construction.

$$\begin{cases} E_{k_x^0}(E_{k_y^0}(k_z^0)) \\ E_{k_x^0}(E_{k_y^1}(k_z^0)) \\ E_{k_x^1}(E_{k_y^0}(k_z^0)) \\ E_{k_x^1}(E_{k_y^1}(k_z^1)) \end{cases}$$

— Bob sends output $k_z^{g(b',b)}$ to Alice and the computation of the garbled gate is complete.

↪ **Generalizing to a circuit** ,

The circuit C is computed gate-by-gate, from the input wires to the output wires, as shown in the following figure 2.11.

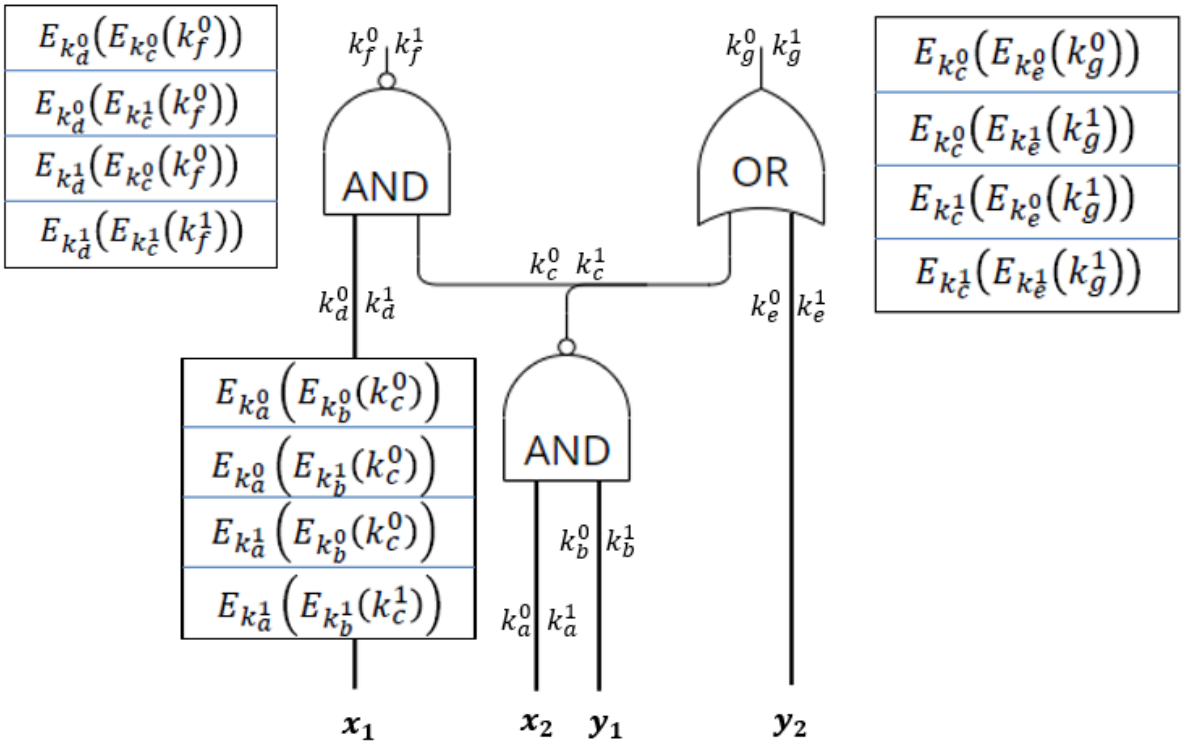


Figure 2.11: Example of Garbled Circuit.

2.6 Outline of Yao's GC Protocol (Generalising Scheme)

Garbling is the process where a Boolean circuit is encrypted [20] . As illustrated in **Figure 2.12**, the initial Boolean circuit representing the function f is given to a **Garbler** Gb which produces three functions :

— **encoding function** En and encoding information e ;

— **decoding function** De and decoding information d ;

— **garbled circuit evaluation function** Ev and garbled circuit C.

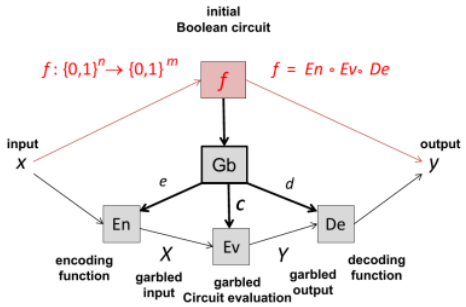


Figure 2.12: Circuit Garbling.

A Garbling scheme is represented as a tuple (Gb, En, Ev, De). As it can be seen from **figure 2.13**, in the Yao protocol, one party, hereafter the sender, builds a garbled circuit C and sends it to the other party, hereafter the receiver. The sender and the receiver then interact in such a way that the receiver obtains the coded inputs $En(x, e)$ and $En(y, e)$. Given these inputs, the receiver then computes the circuit, obtains the output, uses the decoder De and the decoding information d to obtain the actual output and concludes the protocol.

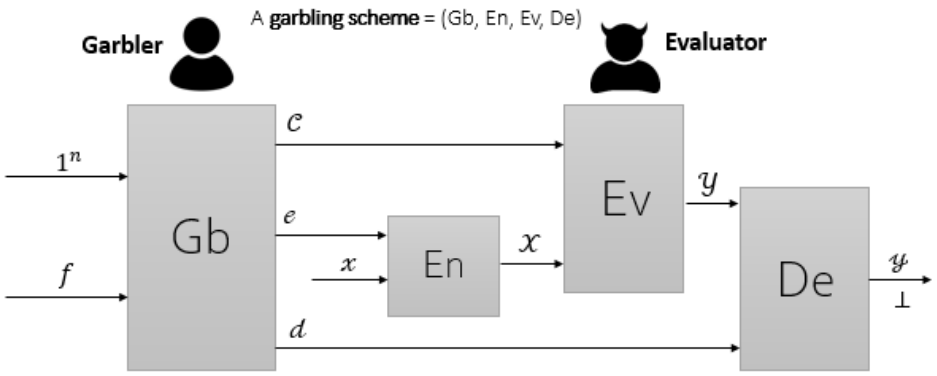


Figure 2.13: Yao's GC Protocol (Generalising Scheme).

The protocol ensure following properties:

— **Privacy** : It ensures that the input is kept private to the parties and that only the encrypted input is revealed, which does not leak any information about the original input.

— **Obliviousness** :The receiver cannot deduce anything from the evaluation output unless the sender sends decoding information d. It ensures output privacy when d is hidden.

— **Authenticity** : It ensures the forgery-proof of the output of Ev.

2.7 Limitations of the Classical Implementation

The feasibility of Yao's Garbled Circuit (GC) approach presented above is strongly dependent on the Oblivious Transfer (OT) cryptographic primitive. In fact, as we will develop in this section, both security level and efficiency of OT protocol set a limit on the security level and computational feasibility of the GC protocol. We will start by exposing some of the limitations of classical OT protocols and then present current research paths that try to mitigate both security and efficiency constraints.

2.7.1 Bellare–Micali OT

As previously stated in section **2.3.2**, the 1-out-of-2 Oblivious Transfer primitive allows a sender S, with two l-bit strings $(x_0; x_1)$, to send x_σ. to a receiver R, when $\sigma \in \{0, 1\}$ is chosen by R. In order to shed some light on the issues related to OT classical implementations we present the Bellare-Micali OT protocol [BM89][21], which makes use of Public Key Diffie-Hellman.

In this protocol, we consider G_q to be a subgroup Z_p^* of with generator g and order q, where p is prime and $p = 2q + 1$. In addition, we assume public knowledge on the value of some constant $C \in G_q$. This constant guarantees that the receiver follows the protocol.

The BM OT protocol is divided into two phases: precomputation and transfer. The first phase defines the resources needed to execute the OT in the second phase. The protocol is depicted in Figure 2.14.

Bellare-Micali

Sender input: m_0, m_1 ($l-$bit strings).
Receiver input: b.

(Precomputation phase)

1. R randomly generates $k \in [1, q]$ and computes g^k.
2. S randomly generates $r_0, r_1 \in [1, q]$ and computes g^{r_0} and g^{r_1}.

(Transfer phase)

3. R sets $PK_b := g^k$. Also, he computes $PK_{1-b} = C \cdot PK_b^{-1}$.
4. R sends both public keys (PK_0, PK_1) to S.
5. S checks if (PK_0, PK_1) were correctly generated by computing their product: $C = PK_0 \times PK_1$.
6. S computes and sends to R the two tuples: $E_0 = (g^{r_0}, H(PK_0^{r_0}) \oplus m_0)$ and $E_1 = (g^{r_1}, H(PK_1^{r_1}) \oplus m_1)$ for some hash function H.
7. R is now able to compute $H(PK_b^{r_b})$ and recover m_b.

Sender output: $\perp$.
Receiver output: m_b.

Figure 2.14: Bellare–Micali oblivious transfer protocol divided into two phases.

2.7.2 Security issues

The BM OT protocol is secure if it complies with both the concealing and the obliviousness property. The former is achieved because the receiver does not send any information to the sender that reveals his input bit choice. The latter relies on the sender's ability to keep his randomly generated elements r_0 andr_1 private. Thus, the obliviousness property is compromised if the receiver is able to compute the discrete logarithm of $g^{r_i}$for $i = 0, 1$ **(discrete logarithm problem)**.

The hardness of the discrete logarithm problem on cyclic groups is the basis of several other important protocols, which makes it crucial to understand its security limits. Nevertheless, it remains to be proven whether, given a general cyclic group G with generator g and order n, a polynomial-time algorithm exists that computes r from g^r , where $r \in Z_r$. Thus, the BM OT protocol's security relies on the assumption that the receiver has limited computational power and is not able to compute the discrete logarithm of a random number.

Although the general discrete logarithm problem is not known to be tractable in polynomial-time, there are specific cases where it is possible to compute it efficiently. This leads to some classical attacks where the structure of the cyclic group considered is not robust enough.

As an **example**, if a prime p is randomly generated without ensuring that $p-1$ contains a big prime p_b in its decomposition, it is possible to use a divide-and-conquer technique [22] along with some other methods to compute the discrete logarithm of a number $a \in G_q$. In this case, the computation time will only depend on the size of p_b . So the smaller p_b is, the faster the algorithm can be. To avoid these types of attacks, safe primes are recommended, that is, $p = 2q + 1$ prime, where q is also prime. However, it is computationally more expensive to find safe primes because they occur less frequently than prime numbers. Beyond the cyclic group structure Otherwise, it is possible to compute the discrete logarithm in an acceptable time. As reported in [23], after one week of precomputation, it is possible to compute the discrete logarithm in a 512-bit group in one minute by using the number field sieve algorithm. So by following this method, after a week-long computation, the receiver would be able to receive both messages m_0 and m_1 of the BM OT protocol in one minute.

In an SMC scenario with the Yao approach, where each OT performed corresponds to one evaluator's input and the chosen group parameters are fixed, the evaluator would be able to obtain the keys corresponding to both 0 and 1 bit, and after some reverse engineering, would be able to discover the garbler's inputs. Therefore, at the expense of efficiency, it is necessary to use big enough prime numbers (2048 bit or larger), for which these classical attacks could not be feasibly implemented.

We have just seen specific examples where it is possible to break the security of an OT protocol using classical techniques. However, it is known that it is possible to break the general discrete logarithm problem with a quantum computer.

2.7.3 The Quantum Technologies Role

In the research literature, there are mainly two approaches to tackle the issue above : the development of protocols with assumptions on the computational power of quantum computers and the development of protocols that make use of quantum technology. The former is known as post-quantum cryptography, and its public-key cryptography protocols are generally more demanding because of the nature of the computational assumptions used. It is also worth stressing that these

computational assumptions are still unproven and have survived just a few years of examination, making the post-quantum approach likely to be attacked in the near future. The latter is known as quantum cryptography [22] and provides solutions without any computational assumption. It has the potential to provide faster protocols but drastically increases the cost of the technological equipment required.

Finally, it is important to note that, in general, quantum protocols do not suffer from intercept now–decipher later attacks because they base their security only on quantum theory. Conversely, this possible threat is always present in protocols based on computational assumptions.

→Following this in-depth study, available in this research article [24], and having seen the importance of developing protocols using quantum technology, the use of Quantum Oblivious Transfer Protocol is admitted by using Quantum Oblivious keys while implementing the Yao'GC Protocol for the development of our Vanet use case.

2.8 Quantum Oblivious Transfer Protocol

Oblivious Transfer (OT) is a cryptographic service in which a sender transfers one of many pieces of information, but it remains oblivious to the receiver that what piece has been transferred. Let Alice and Bob be two agents. A 1-out-of-2 OT service receives strings m_0 and m_1 as input from Alice and b as input from Bob, then outputs m_b to Bob. This is done in a way that Bob gets no information about the other message, and Alice gets no information about Bob's choice b. An OT protocol that uses distributed quantum oblivious keys is proposed as follows in Figure 2.15:

Protocol 1 $\pi_{\mathrm{QK}\to\mathrm{OT}}$

Parameters: Integers $l, r < l/2$, a universal hash function family $\mathbf{F}$ onto $\{0,1\}^r$.

Parties: The sender Alice and the receiver Bob.

Inputs: Alice gets the string m_0 and $m_1 \in \{0,1\}^r$. Bob gets a bit b.

Setup phase:

1. Alice calls an QOKD service, which, from an oblivious key $(k, (\tilde{k}, x))$ of length l, sends k to Alice and $\tilde{k}, x$ to Bob.

Main phase:

2. Bob defines the two $I_0 = \{i|x_i = 0\}$ and $I_1 = \{i|x_i = 1\}$. Then, he sends to Alice the ordered pair $(I_b, I_{b\oplus 1})$.
3. Alice samples $f_0, f_1 \in \mathbf{F}$ and sends to Bob (s_0, s_1), where $s_i = m_i \oplus f_i(k|I_{b\oplus i})$.
4. Bob outputs $m_b = s_b \oplus f_b(k|I_0)$.

Figure 2.15: Quantum Oblivious Transfer Protocol.

2.9 Quantum Random Oblivious Transfer Protocol

In random OT protocol, its not necessary to dedicate input messages for Alice and Bob to start the protocol. In fact, the messages are generated randomly as the output of the QROT protocol. The first two steps of this protocol are identical to the original OT protocol. However, in the third step, the outputs of a universal hash function (which are random) will be used as the sender's messages. Since the receiver will also know what the two sampled hash functions are, he can then obtain one of the random messages without needing further communication with the sender as illustrated in Figure 2.16 :

Protocol 2 $\pi_{\mathrm{OK}\to\mathrm{ROT}}$

Parameters: Integers $l, r < l/2$, a universal hash function family $\mathbf{F}$ onto $\{0,1\}^r$.

Parties: The sender Alice and the receiver Bob.

Setup phase:

1. Alice calls an QOKD service, which, from an oblivious key $(k, (\tilde{k}, x))$ of length l, sends k to Alice and $\tilde{k}, x$ to Bob.

Main phase:

2. Bob defines the two $I_0 = \{i|x_i = 0\}$ and $I_1 = \{i|x_i = 1\}$. Then, he sends to Alice the ordered pair $(I_b, I_{b\oplus 1})$.
3. Alice samples $f_0, f_1 \in \mathbf{F}$ and sets $m_i = f_i(k|_{I_{b+i}})$.
4. Bob outputs $m_b = f_b(k|_{I_0})$.

Figure 2.16: Quantum Random Oblivious Transfer Protocol.

QROT numerical example : In order to have a clear understanding of the Quantum Random Oblivious Transfer (QROT) protocol, we explain the following example: Suppose we have:

Key length $l = 10$

Alice oblivious keys k : 0110010110

Bob oblivious keys: $\tilde{k}$: 0010110111

Bob control keys x : 0110101001 $\rightarrow \begin{cases} I_0 : 03578 \\ I_1 : 12469 \end{cases}$

Note that:

— I_0 correspond to indices of oblivious keys that are known to Bob, while I_1 are indices of oblivious keys that are unknown to Bob.

— Oblivious keys are asymmetric keys, in which one of the parties (Alice) knows all the keys while the other (Bob) only know half of the keys.

Therefore, Bob only can have access to one of the output messages, while Alice, knows all the oblivious keys, can retrieve both output messages. However, since both Alice and Bob's outputs are generated randomly, we call this protocol random oblivious transfer.

Bob, the receiver, can have the choice $b \in \{0, 1\}$ as follows:

$$\text{If Bob choice b} \begin{cases} 0 \rightarrow \text{The ordered pair}(I_0; I_1)\text{is sent to Alice} \\ 1 \rightarrow \text{The ordered pair}(I_1; I_0)\text{is sent to Alice} \end{cases}$$

Suppose Bob choice $b = 1$, Therefore:

Bob output : $m_b = f_b(\tilde{k}|I_0) \xrightarrow{b=1} f_1(\tilde{k}|I_0) = f_1(00111)$

2.10 QOKD Emulator

Quantum Oblivious Key Distribution (QOKD) emulator consists of two programs called : "qokd_emulator_tx.exe" and "qokd_emulator_rx.exe" and are connected to each other using an IP link connection as illustrated in Figure 2.17. The aim of the emulator is to generate and distribute the oblivious keys between parties (Alice and Bob) who may be in far or close distances.

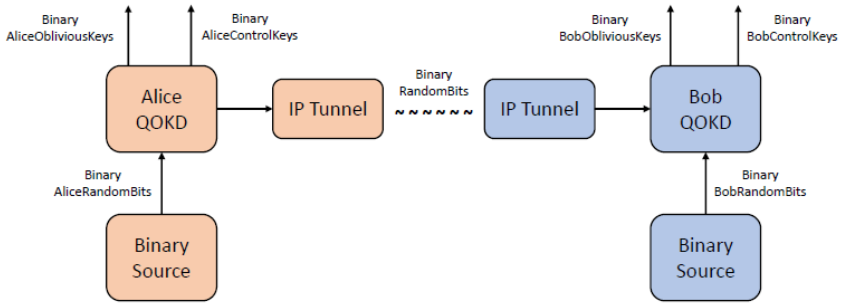


Figure 2.17: Quantum oblivious keys distribution emulator.

→ An implementation of QOKD emulator will be detailed and explained in Chapter 4 when we will implement the Quantum Oblivious Keys in our solution.

Conclusion

This chapter has been dedicated to summarise the bibliographical literature review that we have carried out, which is an essential and mandatory step before starting any project, in which we have defined and presented all the necessary concepts and aspects related to our project. In the following chapter, we analyse the needs of our system and present the conceptual study.

Chapter 3

Analysis and specification of needs

Contents

Introduction

This chapter presents a general analysis of the system to be developed. First, we will present the architectural design of our future system and the different functionalities it proposes. Then, we will highlight the different functional and non-functional requirements of our system. Finally, we will try to show the possibilities of the system and the users' needs in use case diagrams and a static overview through a class diagram.

3.1 System Model

Our project consists in the construction of an infotainment application for VANET drivers to find and recommend places/services (hotel, restaurant, gas station,...) which are based on the user's interests such as: type of cuisine, recommendation level, local traffic situation, shortest way to a certain POI. For our part, we will focus on a single type of service that consists of finding the nearest restaurant based on the driver's location when arriving in an unfamiliar place, our application will be a reliable service to recommend potential places of interest.

For this purpose, we will apply the MPC concept to compute data and arrive at a mutually desired result without the parties having to disclose their private data (which is the driver's location in our use case) by implementing the Yao'GC protocol for 2-PC. A general architecture is shown in **figure 3.1** in which we try to present the flow of our project.

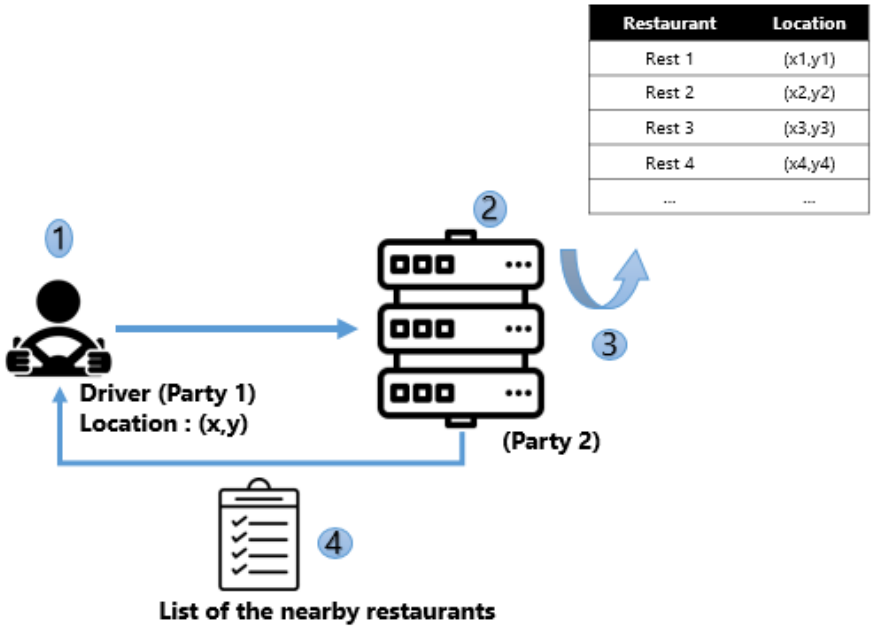


Figure 3.1: The system model.

1. a driver enters his/her private location and chooses the preferred service ;
2. party-2 computes a function to find the nearest service according to the driver's preferences using Yao's Protocol ;
3. party-2 has a list of restaurant for that specific location, It will be used to calculate the distance from the driver to each location ;
4. party-2 outputs a list of recommended restaurants based on the minimum distance.

3.2 Capture of Needs

Requirements analysis is the process of determining what users expect from a new or modified product. These characteristics, called requirements, must be quantifiable, relevant and detailed. In software engineering, these requirements are often referred to as functional specifications. Requirements analysis is an important aspect of project management that enables the success of a system or software project to be assessed. To do so, it is important to make a detailed and clear description of the users' expectations and the functionalities that the system will provide.

There are two main types of system requirements that need to be gathered while we are working on a software project. Following are the functional and non-functional requirements of our project which will present a common understanding between the specialists and the end-users of the system in order to to avoid an unsatisfying project.

3.2.1 Functional needs

It can be most simply defined as: Something the system must do in response to different inputs and what it must output.If the system does not meet a functional requirement it will fail. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

In this context, our application must meet the following functional requirements:

— receive the driver's location and find all the restaurants that are nearby in that area without disclosing the privacy of the driver's location ;

— compute a distance function to find the nearest service according to the driver's location ;

— recommend some places that are based on the interests of the user ;

— use Yao's GC protocole while performing the computation and arriving at a mutually desired result without requiring parties to divulge their private data ;

— only the authorised party will get the result of the calculation, the potential restaurant in proximity.

→ To fulfil these functional requirements, there are a number of aspects that must be respected otherwise our application does not make much sense:

Data confidentiality : data confidentiality requires that the data must be invisible to malicious servers and users. By taking advantage of the authentication algorithm, we identify malicious users to prevent data leakage.

Flexibility : As time goes by, users' interests change. If the user's interests are not updated regularly, the accuracy of the recommendation algorithm will be seriously affected. Therefore, the system should update data in a flexible way.

Dynamic change : Our scheme should supports the dynamic change of users' numbers .

Accuracy : Users expect to get accurate recommendations using the recommendation algorithm. At the beginning of the data processing, the accuracy of the personal data analysis is particularly important, as the deviation of the data analysis will affect the use of the filtering algorithm.

Timeliness : In VANETs, the response time of the algorithm should be reduced as much as possible. Therefore, our scheme should reduce the time overhead to boost the speed of the algorithm.

3.2.2 Non-functional needs

Non-functional requirements focus on how the system goes about performing a specific function. At first sight they may be considered less important than functional requirements, but both have a role to play in delevering a good system. They are also called non-behavioral requirements. The main non-functional requirements of our application can be summed up in the following points:

— Code must be clear to allow future evolutions or improvements ;

— **Availability**: the application must be available for use by any user ;

— **Ergonomy**: the application must provide a user-friendly and simple interface for all types of users, as it is the first contact of the user with the application and through this one will discover its functionalities ;

— **Reliability**: the data provided by the application must be reliable ;

— An **open and evolutive solution**: the application can be enhanced by adding other modules to ensure the flexibility, scalability and adaptability of the solution .

3.3 Needs Modeling

Before starting a software project and jumping into the code, we should first structure our ideas and organise them well, then prepare the implementation by setting up its modules and stages in a coherent way. This approach is called modelling: it is a kind of simplification of the project which will specify the data structures and the behaviour of the system to provide at the end a guide which will help the understanding and the construction of the project. For this project, it was decided to use the UML methodology for the modelling of our application.

3.3.1 Unified Modeling Language (UML)

UML is a standardised modelling language that consists of an integrated set of diagrams, developed to help system and software developers illustrate quantifiable aspects of a system that can be described visually, such as relationships, behaviour, structure and functionality, as well as for modelling business and other non-software systems.

In the following section we present the modelling of the system requirements through a **Use Case Diagram**, and a static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects through a **Class Diagram** .

3.3.2 Use Case Diagram

Use case diagrams consist of 4 main objects :

— **Actor** : is any entity that interacts with the given system. It can be a person, an organisation or an external system.

— **Use case** : Describes a functionality of the system that is seen by an actor and is useful to him. It represents a service provided by the system, usualy drawn as an oval and named with the function.

— **System** : is used to define the scope of the use case and drawn as a rectangle.

— **Relationships** : is a connection between model elements grouped into categories : Associations, Generalizations, Include relationships and Extend relationships.

3.3.2.1 Identification of the actors

An actor specifies a role played by a user or any other system that interacts with the subject. in Table 3.1 we identify the main actors involved in our system:

Table 3.1: Actors Identification.

Actors	Role
Party-1 (=VANET Driver)	— Login into the application ; — Select a service (e.g.car parks, petrol stations, shopping centres, hotels, restaurants), according to his/her interests; — Select a preference according to the chosen service ; — Identify its position and send it privately to Part-2.
Party-2 (=Serveur/RSU)	— Identify user; — Authentificate user; — Receive and save the driver's current location for use in the calculation process; — Map the driver's location to find nearby services based on the user's interests and preferences; — Compute a distance function by implementing Yao's GC Protocol ; — Send back the computation result encoded to only authorised Party (VANET Driver).

3.3.2.2 Global Use Case Diagram

Here as shown in figure 3.2 a general overview of our use case diagram with the main 4-objects as described above :

Figure 3.2: Global use case Diagram

→ This use case diagram presents a general view of the system we want to develop, but for our part we are limited to one service which is to find the nearby restaurants to the driver's location.

3.3.2.3 Use Case Description

In order to better understand the functional requirements, we present their specification, through detailed use case diagrams and textual descriptions of the corresponding scenarios.

Figure 3.3 shows the "Select Service" use case and the corresponding textual description is presented in the table 3.2.

- Refinement of "Select Service" use case

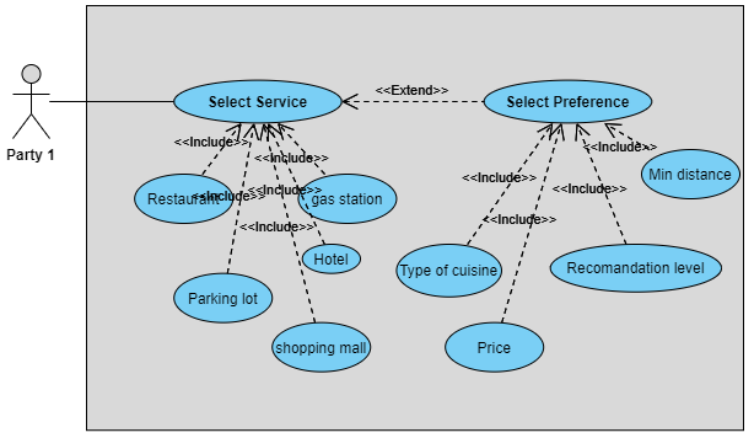


Figure 3.3: "Select Service" use case

Table 3.2: Text description of the scenario "Select Service" .

Use Case	**"Select Service"**
Brief Description	Party 1 can select a service from the list of services offered,and depending on its selection, it can also select a preference according to the service.
Actors	Party 1
Preconditions	Login to the application.
Postconditions	Service and preferences successfully selected.
Nominal Scenario	— Party-1 login the application ; — the home interface is displayed and a list of available services is shown ; — Party-1 chooses a service based on its need ; — Party 1 chooses a preference based on the service selected.
Alternative Scenario	— The selection made by the driver is not valid ; — The System does not proceed and an error message is displayed ;

Figure 3.4 shows the "Send Private location" use case and the corresponding textual description is presented in the table 3.3.

- Refinement of "Send Private location" use case

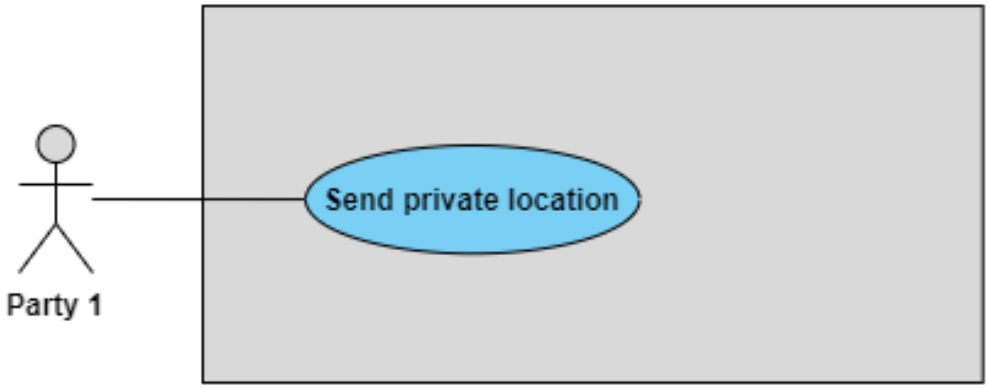


Figure 3.4: "Send Private location" use case

Table 3.3: Text description of the scenario "Send Private location" .

Use Case	**"Send Private location"**
Brief Description	Party 1 identifies its position and transmits it to part 2 in encoded form.
Actors	Party 1
Preconditions	Service and preference selected .
Postconditions	The driver's location is sent to part 2 successfully and without revealing its actual content.
Nominal Scenario	— Party-1 indicate his location ; — Party-1 encode the location information ; — Party-1 send the encoded information to Party-2 ;
Alternative Scenario	If the position entered by party-1 is not in conformity with the format recognised by the system, an error message is displayed and the user is redirected to the home page.

Figure 3.5 shows the "Identify Nearby services" use case and the corresponding textual description is presented in the table 3.4.

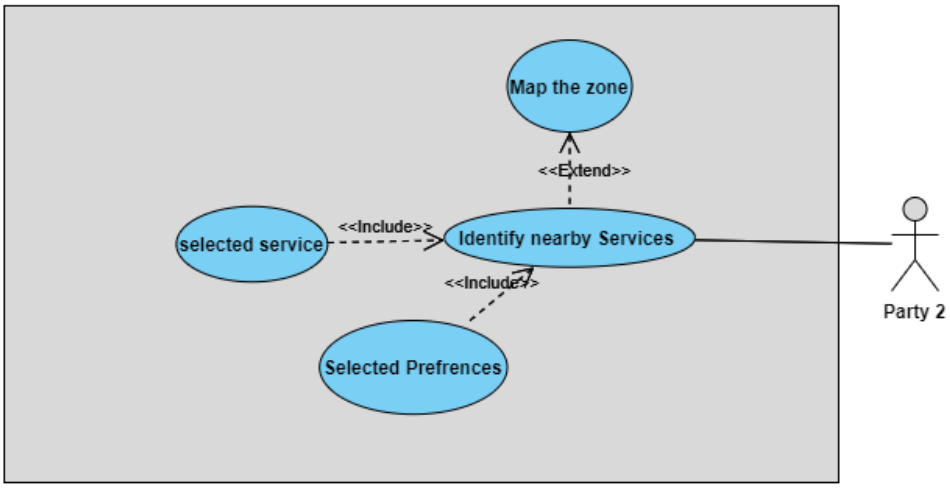


Figure 3.5: "Identify Nearby services" use case

- Refinement of "Identify Nearby services" use case

Table 3.4: Text description of the scenario "Identify Nearby services" .

Use Case	**"Identify Nearby services"**
Brief Description	Party-2 maps the area near the driver's location and identifies a list of service locations based on the driver's choice and preference.
Actors	Party 2
Preconditions	driver's location received and saved
Postconditions	a list of nearby services is available
Nominal Scenario	— Party-2 identify services nearby the driver's location ; — Party-2 organize them into a list ;
Alternative Scenario	if party-2 is not able to find services in the nearby area according to the driver's location, it will search in a wider area, such as the whole city.

Figure 3.6 shows the "Compute distance function"" use case and the corresponding textual description is presented in the table 3.5.

- Refinement of "Compute distance function" use case

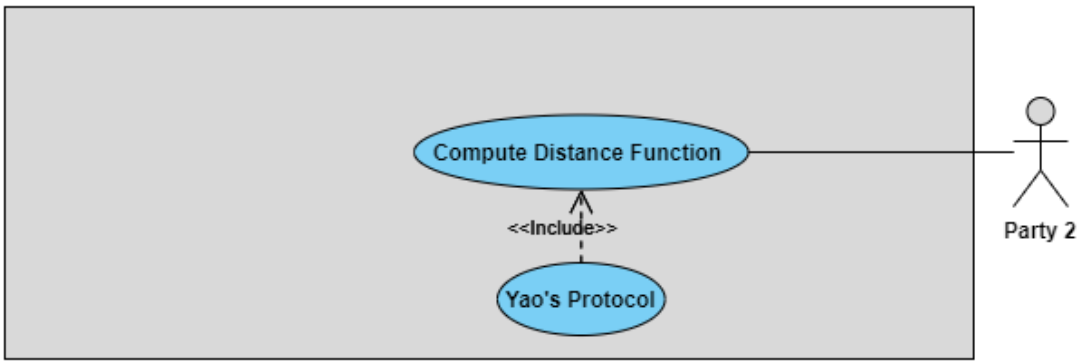


Figure 3.6: "Compute distance function" use case

Table 3.5: Text description of the scenario "Compute distance function" .

Use Case	**"Compute distance function"**
Brief Description	Party-2 implements the Yao GC protocol for copmuting a distance function to find the shortest path to a specific service.
Actors	Party-2
Preconditions	Driver's location saved and a list of nearby services is identified.
Postconditions	the distance function is computed
Nominal Scenario	— Party-2 compute a distance function between each location in the list and the driver's location ; — Party-2 identify the Minimum distance among all calculated distances .
Alternative Scenario	— Protocol fail ; — The system displays an error message.

Figure 3.7 shows the "Save Driver's location" use case and the corresponding textual description is presented in the table 3.6.

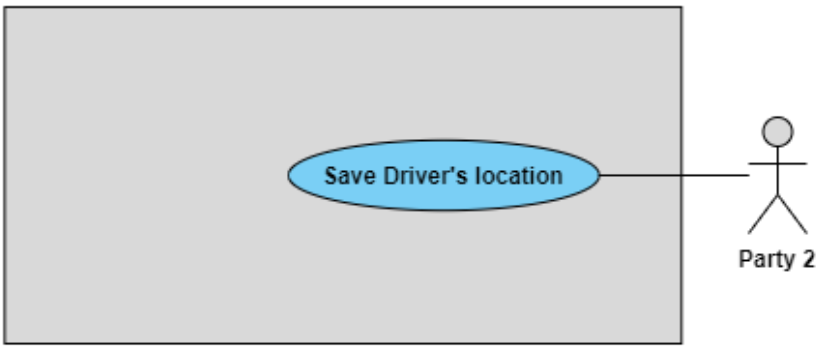


Figure 3.7: "Save Driver's location" use case

- Refinement of "Save Driver's location" use case

Table 3.6: Text description of the scenario "Save Driver's location" .

Use Case	"Save Driver's location"
Brief Description	Party-2 saves the driver's position for later use in the distance calculation process.
Actors	Party 2
Preconditions	The driver's location is sent by party-1
Postconditions	Driver's location is sent
Nominal Scenario	Party-2 recirve driver's location and save the location information.
Alternative Scenario	If the connection fails and registration has not been completed, an error message is displayed.

Figure 3.8 shows the "Send back result to Party 1" use case and the corresponding textual description is presented in the table 3.7.

- Refinement of "Send back result to Party 1" use case

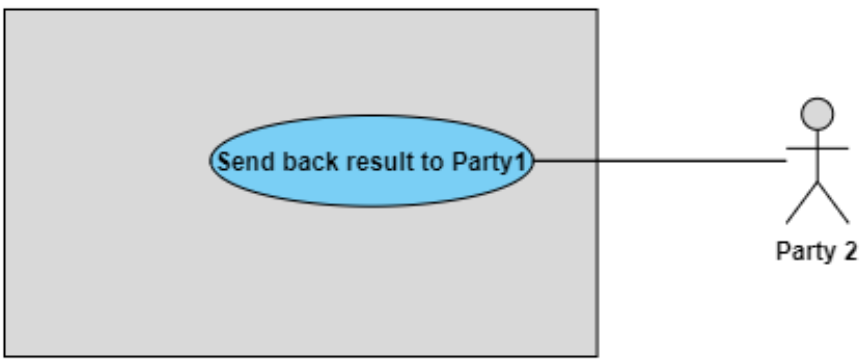


Figure 3.8: "Send back result to Party 1" use case

Table 3.7: Text description of the scenario "Send back result to Party 1" .

Use Case	**"Send back result to Party 1"**
Brief Description	After calculating the distance function and identifying the nearby service, this information should be sent to the driver in coded form so that only the authorised party can read it.
Actors	Party-2
Preconditions	the distance function computation is completed and the nearby service is identified.
Postconditions	the service nearby is forwarded to only authorised Party.
Nominal scenario	— Party-2 encode the result information ; — Part 2 returns the encrypted information so that only the authorised party can see the information using the Yao protocol.
Alternative Scenario	an error message will be displayed if something goes wrong with the Yao protocol.

3.3.3 Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. We use the class diagram to visualize, describe, document various different aspects of the system, and also construct executable software code.

In the following figure 3.9 we shows the attributes, classes, functions, and relationships to give an overview of our software system.

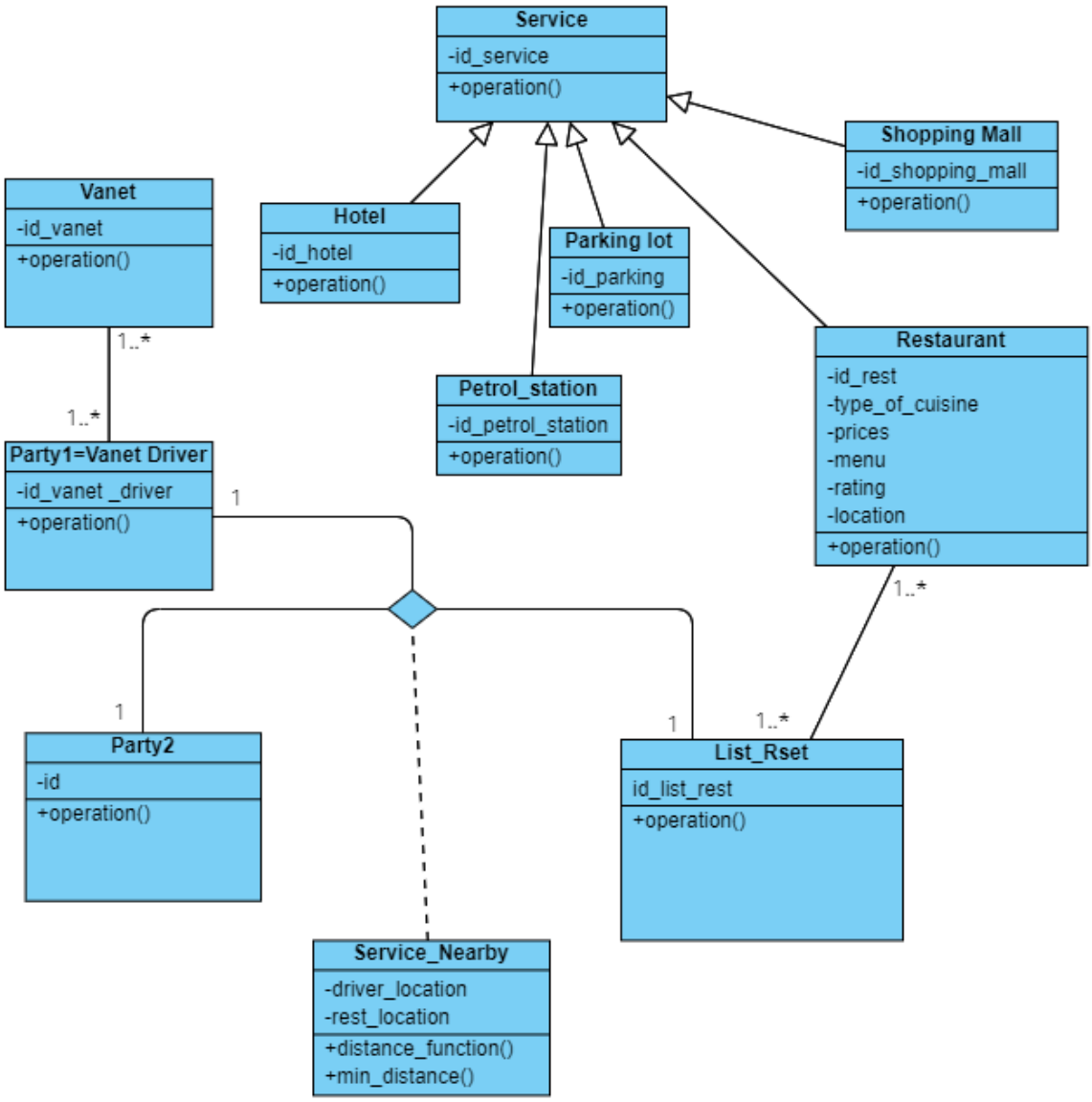


Figure 3.9: Class Diagram

3.4 Project Life Cycle

The most suitable model for our project is the spiral model. It focuses on risk management. The risks to be taken into account are very diffrent: unavailability of hardware or certain software, change in requirements, time overrun, cost overrun, underestimated project size and change in technology.

The spiral model is used to focus on successive prototypes of the application. The following figure 3.10 illustrates the spiral cycle of our project, from determing objectives, alternatives and constraints to the implementation of the final functional prototype phase.

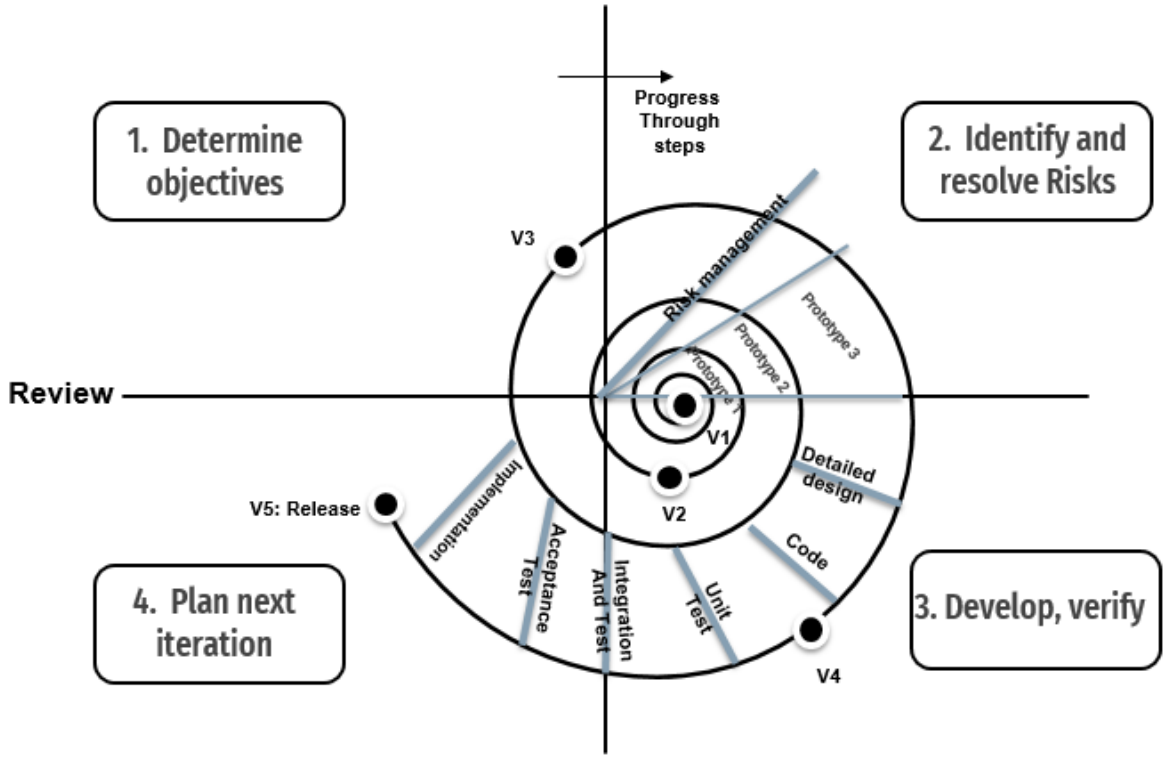


Figure 3.10: Spiral Model

— **V1 :** High level python code ;

— **V2 :** Integration of the high level python code into the MP-SPDZ library ;

— **V3 :** Implementation of Yao's GC Protocol ;

— **V4 :** Integration of Quantum Oblivious Keys ;

— **V5 :** the final release with Carla Demo ;

Conclusion

In this chapter, we have detailed the different conceptual views of the application to be implemented through the necessary UML models. Then we have defined the life cycle adopted for this project. This step is essential for the implementation phase, which is the focus of the next chapter.

Chapter 4

Implementation and Results

Contents

Introduction

In this chapter we describe in details how to develop general Secure Multiparty Computation application assisted with quantum primitives. We start by explaining how to set the MP-SPDZ Library first with a simple classical implementation and then we evolve to a more complex implementation. Finaly, we will end up by the implementation of Carla Simulator, which provides an environment where the protocol can be executed.

4.1 Multi Party Computation - MP/SPDZ implementation

MP-SPDZ is a software to benchmark various secure multi-party computation protocols such as SPDZ, MASCOT, Overdrive, BMR garbled circuits, Yao's garbled circuit, to name a few. It extends SPDZ-2 to 34 MPC protocol variants, all of which can be used with the same high-level programming interface based on Python.This considerably simplifies comparing the cost of different protocols and security models.

The protocols cover all commonly used security models (honest/dishonest majority and semi-honest/malicious corruption) as well as computation of binary and arithmetic circuits (the latter modulo primes and powers of two). The underlying primitives employed include : Secret Sharing, Oblivious Transfer, Homomorphic Encryption and Garbled Circuits.

4.1.1 MP-SPDZ Architecture

For the actual computation, the software implements a virtual machine that executes programs in a specific bytecode as illustrated in Figure 4.1 .

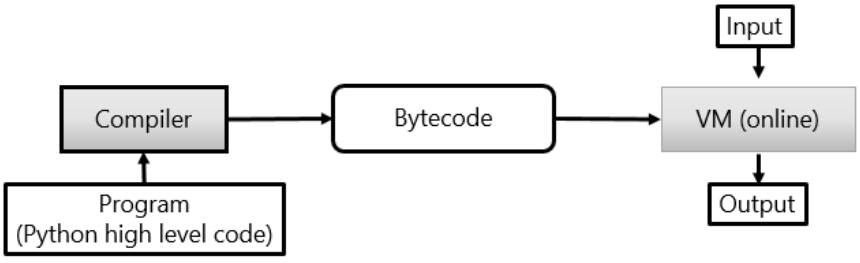


Figure 4.1: MPC compiler and VM (runtime enviromment)

The programs containing the functions to be evaluated by the protocol are written in high-level Python code, which is then used to generate a specific bytecode. Such code can be generated from high-level Python code using a compiler that optimizes the computation with a particular focus

on minimizing the number of communication rounds (for protocol based on secret sharing) or on AES-NI pipelining (for garbled circuits). The software uses two different bytecode sets, one for arithmetic circuits and one for boolean circuits.

A pictorial representation of the MP-SPDZ repository is in Figure 4.2, it shows how the SPDZ compiler takes the high-level python-like program as input and outputs corresponding bytecodes. Afterward, the virtual machine is able to perform MPC protocols, by interpreting these bytecodes.

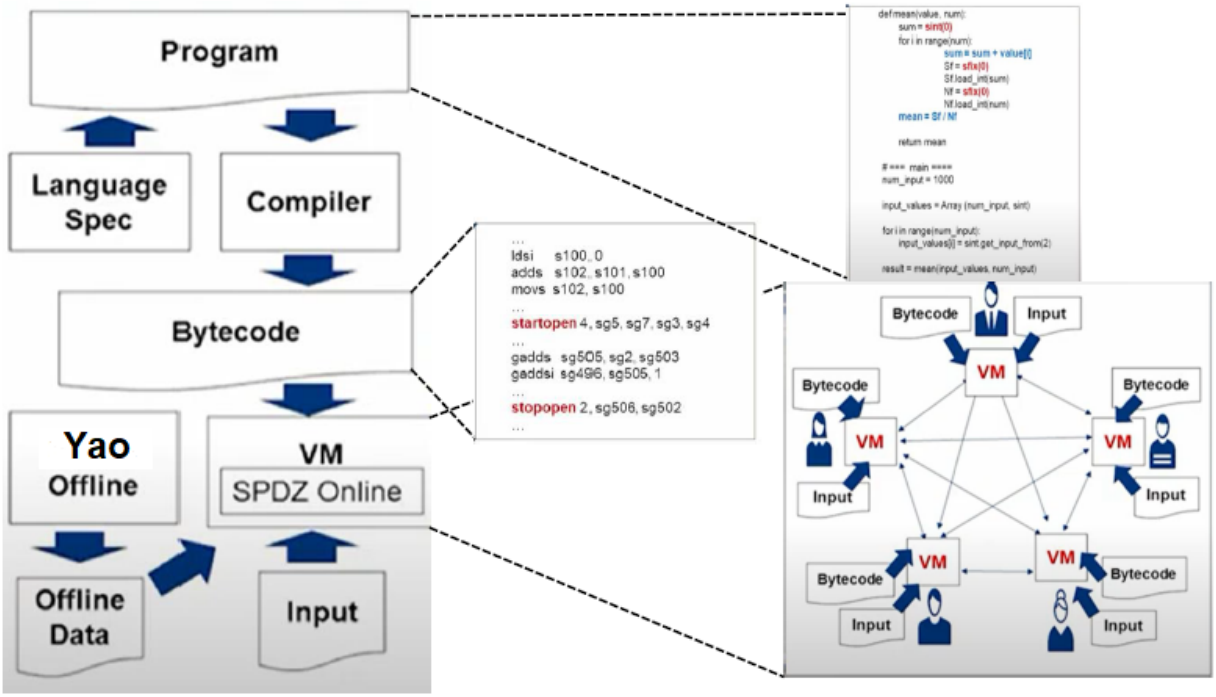


Figure 4.2: MP-SPDZ Architecture

Then,the generated executable (yao-party.x), as well as the predefined circuit should be distributed among all parties in the local network. Afterwards, the circuit and the private inputs of each party is feeded to the yao-part.x to perform secure multiparty computation as shown in the folowing figure 4.3 .

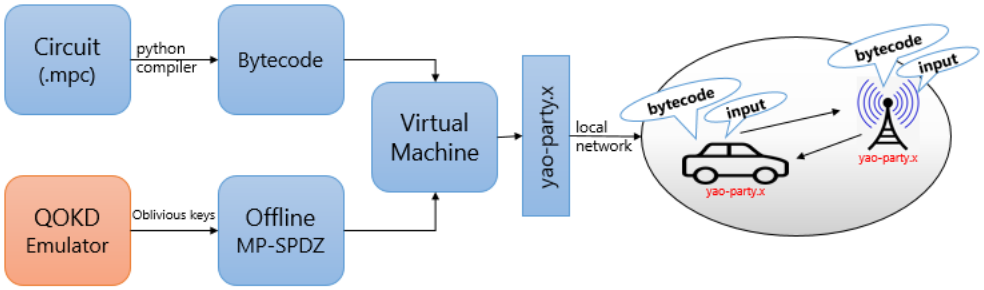


Figure 4.3: Generation of executable yao-party.x

4.1.2 MP-SPDZ Requirements

MP-SPDZ framework requires either a Linux distribution originally released 2014 or later (glibc 2.17) or macOS High Sierra or later as well as Python 3 and basic command-line utilities.

The following is a list of requirements to properly install the MP-SPDZ framework by following the step-by-step instructions available at the MP-SPDZ documentation [25] :

— **GCC 5** or later or **LLVM/clang 5** or later. clang is most recommanded because it performs better.

— For protocol using oblivious transfer, **libOTe** is required with the necessary patches but without SimplestOT. The easiest way is to run make libote, which will install it as needed in a subdirectory. libOTe requires CMake of version at least 3.15, which is not available by default on older systems such as Ubuntu 18.04. but we can can run make cmake to install it locally.

— **MPIR library**, compiled with C++ support (with using flag –enable-cxx when running configure).We can use `make -j8 mpir` to install it locally as an alternative solution.

— **libsodium** library

— **OpenSSL**

— **Boost.Asio** with SSL support (libboost-dev on Ubuntu)

— **Boost.Thread** for BMR (libboost-thread-dev on Ubuntu)

— **Python 3.5** or later

— x86 or ARM 64-bit CPU

— NTL library for homomorphic encryption

→ To properly install the MP-SPDZ framework, we first install the Linux distribution in a VMware-based virtual machine, then adjust the settings according to our preferences and needs and then start installing all the necessary requirements with the matching versions.

4.1.3 MP-SPDZ Compilation

1. Edit `CONFIG` or `CONFIG.mine`, used to configure the parameters and initial settings of the software according to its needs. But we need to remember to run make clean first after changing in those files. Some of the changes could be :

- To benchmark online-only protocols or Overdrive offline phases, we need to add the following line at the top: `MY_CFLAGS = -DINSECURE`
- For homomorphic encryption with $GF(2 \wedge 40)$, we need to set USE_NTL = 1 .
- If we intend to run on a different CPU than compiling, we might need to change the ARCH variable in `CONFIG` or `CONFIG.mine` to `-march=<cpu>`.

2. We run `make` to compile all the software (also we can use the flag `-j` for faster compilation using multiple threads).

Figure 4.4 shows what an MP-SPDZ framework looks like after all requirements have been installed and the software compiled.

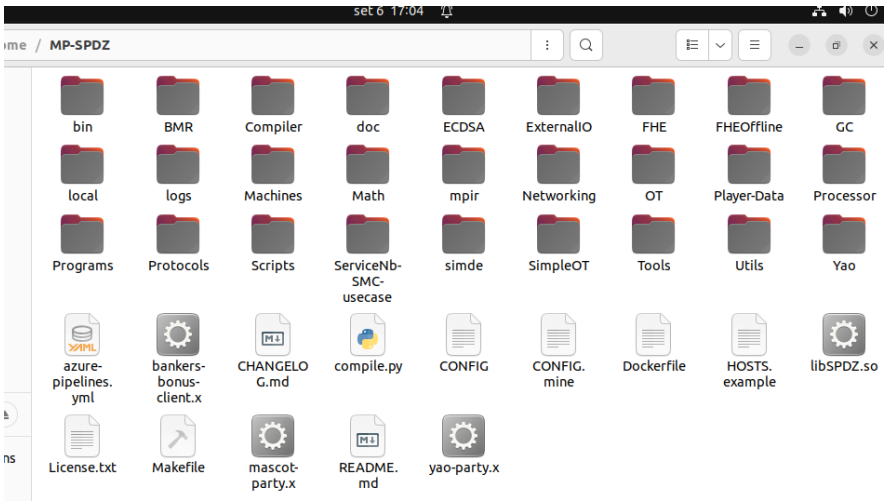
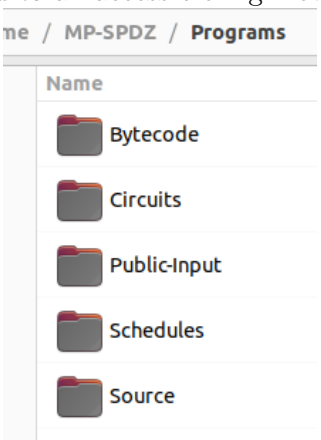


Figure 4.4: MP-SPDZ Framework.

→ As we can see from the figure, MP-SPDZ covers from 2 to 30 MPC protocol variants, we can also see primitives like secret sharing, OT and GC employed. All implemented protocols are coupled to an accessible high-level interface.



in `Programs/Source/` we place our high level `.mpc` program, the generated bytecode and circuit are found in `Programs/Bytecode/` and `Programs/Circuits/` , and if we need to use dynamically read public inputs, they should be placed in `Programs/public-input/` .

Figure 4.5: MP-SPDZ/Programs Directory.

4.2 Yao'GC implementation

we start by transforming the use case we have detailed in the previous chapters into a function that takes input from vanet driver (Party 1) and input from Party 2, a computation is performed and then only the party concerned by the result can read its content.

What we are going to do next is to start writing our `.mpc` file as shown in Figure 4.6 containing the circuit(=computation function) that should be placed in `Programs/Source/` .

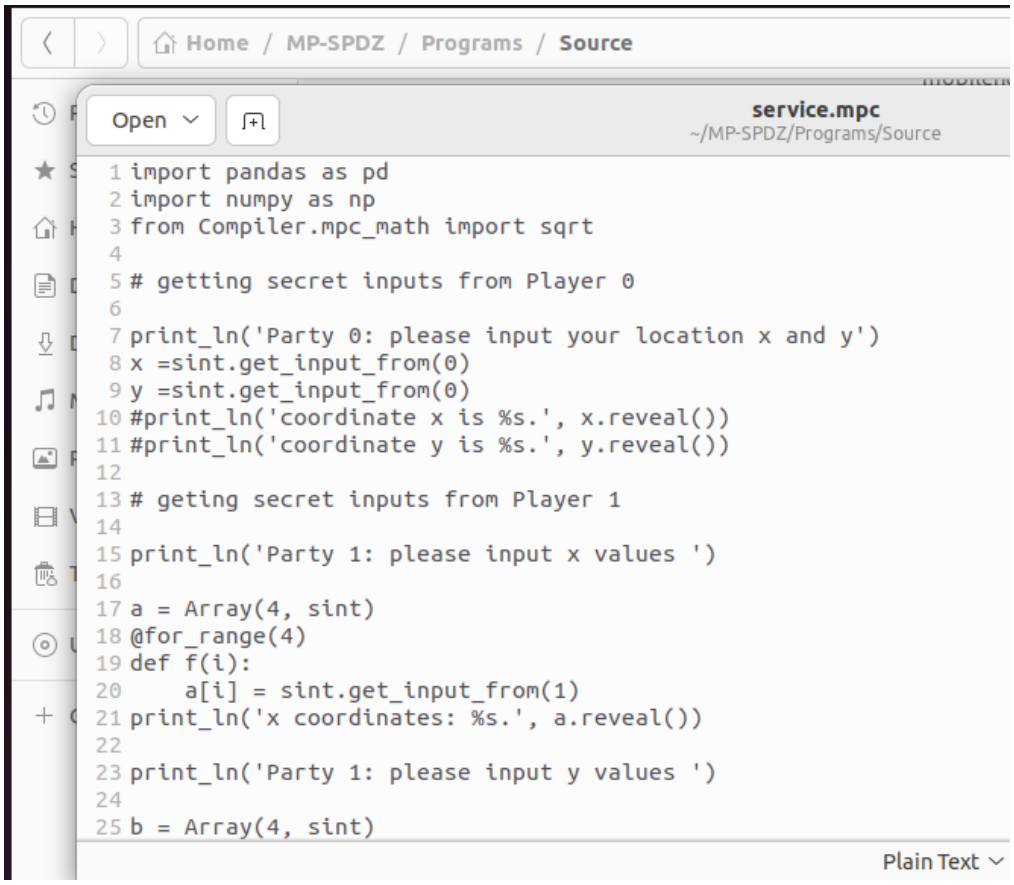


Figure 4.6: .mpc file

We can See `Programs/Source/` for some example MPC programs, in particular tutorial.mpc. Furthermore, we could read the Docs hosts [26] a more detailed reference of the high-level functionality extracted from the Python code in the Compiler directory as well as a summary of relevant compiler options in order to learn how to write and compile our function properly and coorrectly.

Some of the methods, functions and basic types specific to the MP-SPDZ framework used in this `.mpc` file are explained subsequently .

Basic types and Container types

These include basic types used in our `.mpc` file such as secret integers or floating-point numbers as shown int table 4.1 and container types as shown in table 4.2. A single instance of the former uses one or more so-called registers in the virtual machine while the latter use the so-called memory. For every register type, there is a corresponding dedicated memory.

Table 4.1: Basic Types.

Sint	Secret integer in the protocolspecific domain.
Cint	Clear integer in same domain as secure computation (depends on protocol).
Sfix	Secret fixed-point number represented as secret integer,by multiplying with $2 \wedge f$ and then rounding.
Sfloat	Secret floating-point number.
regint	Clear 64-bit integer.

Table 4.2: Container types .

Array	Array accessible by public index..
Matrix	Matrix.
MultiArray	Multidimensional array. .

- `get_input_from(player, n_bits=None)` : is a method to get a secret input from a player and take in as a parameter:

  player number : int

- `print_ln(s='', *args)`: Print line, with optional args for adding variables/registers with `%s`. By default only player 0 outputs, but the I commandline option changes that. takes in as parameters:

  s : Python string with same number of `%s` as length of `args`

  args : list of public values (regint/cint/int/cfix/cfloat/localint)

- `print_ln_to(player, ss, *args)` : Print line at a specific player only. takes in as parameters:

  player : int

  ss : Python string

  args : list of values known to player

- `for_range(start, stop=None, step=None)` : instead of using Python loops (`for`) which are unrolled at compile-time, resulting in potentially too much virtual machine code ,we consider using `for_range()` or similar, which will replace simple loops by an optimized version. takes in as parameter:

start/stop/step : regint/cint/int

- `reveal(*args, **kwargs)` : Reveal secret bit register vectors and copy result to clear bit register vectors (it simply return relevant clear type).
- `reveal_to(player)` : Reveal secret value to a specific player. takes in as parameter:

  player : public integer (int/regint/cint)

4.3 Yao'GC Compilation

After we have written our function in the `.mpc` file we go on to compile our circuit.

→ To compile the Yao protocol for two parties on one machine, we apply the following steps:

1. First compile the virtual machine: `make -j 8 yao`
2. Then we compile the high-level program: `./compile.py -B <integer bit length> <program>`

   •The option **-B** indicates Binary circuits : the computation domains.

   •The **integer bit length** can be any number up to a maximum depending on the protocol.All protocols support at least 64-bit integers.so we set it by default to 64.
3. Then we run the protocol with two parties on one machine in separate terminals :

   Garbler: `./yao-party.x [-I] -p 0 <program>`

   Evaluator: `./yao-party.x [-I] -p 1 -h <garbler host> <program>`

   •The option **-h** indicates the hostname of players, and **-p** the player we are running.

   •The option **-I** activates interactive input, otherwise inputs are read from :

`Player-Data/Input-P<playerno>-0`.

→ here we show the result from running compilation of our circuit following the stepes described above:

[1.] First, as shown in Figure 4.7 we start by compiling the virtual machine:

```
oumayma@oumayma-virtual-machine:~/MP-SPDZ$ make -j 8 yao
clang++ -o Machines/yao-party.o Machines/yao-party.cpp -march=native -I./local/i
nclude -I./local/include -I./local/include -DFULL_GATES -I./local/include -I./lo
cal/include -g -Wextra -Wall -O3 -I. -pthread    -DUSE_GF2N_LONG '-DPREP_DIR="Pl
ayer-Data/"' '-DSSL_DIR="Player-Data/"'  -std=c++11 -Werror -fPIC -MMD -MP -c
clang++ -o OT/OTTripleSetup.o OT/OTTripleSetup.cpp -march=native -I./local/inclu
```

Figure 4.7: Compiling virtual machine.

[2.] Second, compiling the high-level program as shown in Figure 4.8:

```
oumayma@oumayma-virtual-machine:~/MP-SPDZ$ ./compile.py -B 64 service
Default bit length: 64
Default security parameter: 40
Compiling file /home/oumayma/MP-SPDZ/Programs/Source/service.mpc
Compiled 100000 lines at Fri Sep  2 10:50:33 2022
Compiled 200000 lines at Fri Sep  2 10:50:36 2022
Compiled 300000 lines at Fri Sep  2 10:50:38 2022
WARNING: Order of memory instructions not preserved, errors possible
WARNING: Order of memory instructions not preserved, errors possible
WARNING: Order of memory instructions not preserved, errors possible
WARNING: Order of memory instructions not preserved, errors possible
WARNING: Order of memory instructions not preserved, errors possible
WARNING: Order of memory instructions not preserved, errors possible
WARNING: Order of memory instructions not preserved, errors possible
Writing to /home/oumayma/MP-SPDZ/Programs/Schedules/service.sch
Writing to /home/oumayma/MP-SPDZ/Programs/Bytecode/service-0.bc
Program requires at most:
         128 bit inputs from player 0
         512 bit inputs from player 1
      391607 bit triples
       10241 virtual machine rounds
oumayma@oumayma-virtual-machine:~/MP-SPDZ$
```

Figure 4.8: Compiling high-level program.

→ During compilation, some information are displayed, such as the length of the input bits of party 1 and 2 and the number of rounds of the virtual machine, which means that when the bytecode is generated from the high-level Python code, the compiler optimises the calculation according to the number of communication rounds and displays the number of rounds afterwards.

[3.] Finally, the compilation of the Yao protocol with two parts "Garbler" and "Evaluator" on separate terminals : Party 0 first enters its position as an (x=long,y=lat) coordinate. This information is kept private.

Note: Throughout the process this information is kept secret by the protocol and if we can see this information from the figure 4.9 below it is because I have used the `reveal()` method to see the driver's input to better understand the execution process.

```
oumayma@oumayma-virtual-machine:~/MP-SPDZ$ ./yao-party.x -I -p 0 service
Party 0: please input your location x and y
Please enter 2 numbers:
15
63
Thank you
```

Figure 4.9: Party 0 inputs its location.

Then party 0 gives as input a list of restaurants, in the same coordinate format (x=lon,y=lat) in the proximity of the driver's location after having mapped this area as shown in figure 4.10.

```
oumayma@oumayma-virtual-machine:~/MP-SPDZ$ ./yao-party.x -I -p 1 service
Please enter 1 numbers:
10
Thank you
Please enter 1 numbers:
45
Thank you
Please enter 1 numbers:
12
Thank you
Please enter 1 numbers:
19
Thank you
Please enter 1 numbers:
5
Thank you
Please enter 1 numbers:

45
Thank you
Please enter 1 numbers:
56
Thank you
Please enter 1 numbers:
-6
Thank you
```

Figure 4.10: Party 1 inputs a set of restaurant locations.

→ This list of values could be read from a dedicated csv file or a database, but as a simple assumption, we manually enter this list with random numbers.

Finally, the end result for Party 0 as shown in figure 4.11:

```
Party 1: please input x values
x coordinates: [10, 45, 12, 19].
Party 1: please input y values
y coordinates: [5, 45, 56, -6].
distance list : 63.1404
distance list : 35.4485
distance list : 7.62082
distance list : 73.2557
 REST LIST: Name and corresponding distance
 [ ID/RESTA: 50 , 63.1404]
 [ID/RESTB: 20 , 35.4485]
 [ID/RESTC: 10 , 7.62082]
 [ID/RESTD: 90 , 73.2557]
sorted Matrixs
 [7.62082, 10]
 [35.4485, 20]
 [63.1404, 50]
 [73.2557, 90]
3 places to eat near you:
[7.62082, 10]
[35.4485, 20]
[63.1404, 50]
Instruction statistics:
Receiving directly 0.009552 MB in 14 rounds, taking 12.1747 seconds
Sending directly 25.064 MB in 30 rounds, taking 0.0265003 seconds
Time = 84.9262 seconds
Data sent = 25.064 MB
oumayma@oumayma-virtual-machine:~/MP-SPDZ$
```

Figure 4.11: result revealed to Party 0.

→ 3 recommendations of restaurants near the driver's location are displayed with the corresponding distance and the respective restaurant ID. Other information are also calculated and available in the statistics part of the instructions, such as the quantity of data received and sent per number of virtual machine rounds, the time taken to perform this computation and the data sent.

4.4 Yao Implementation - Integrating Quantum Primitives

In this section, we explain the integration of QOKD emulator which was introduce in section **2.10** (Chapter 2) to MP-SPDZ framework.

The OT protocol used previously in MP-SPDZ framework is a classical protocol derived from the Diffie-Hellman key exchange and whose security relies on the Computational Diffie-Hellman assumption. Note that this protocol has of two phases: the first one consists of a random OT, where, the sender does not choose her input messages; in the second phase, the sender uses the data generated during the random OT to encrypt his messages and send them to the receiver (who will only be able to decrypt one of them).

→ The easiest way to integrate a quantum OT primitive (and the one we will be implementing) is by replacing this base OT with the desired quantum OT protocol.

We implemented this protocol and integrated it in the original Yao implementation. The operations done by the sender and the receiver are defined in `qot_sender.c` and `qot_receiver.c`, respectively as shown in Figure 4.12, and in the corresponding header files we define a sender structure and a receiver structure, which will hold some of the data used by each player throughout the protocol.

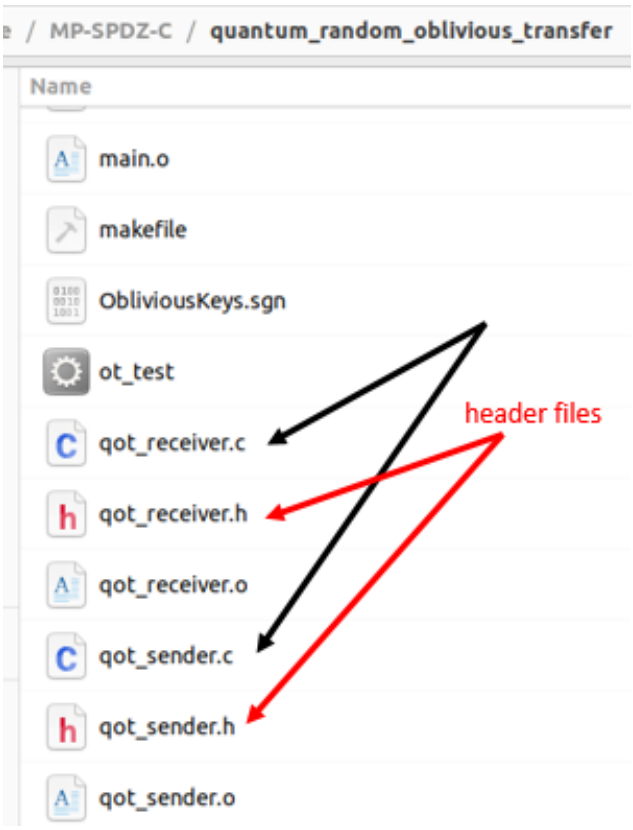


Figure 4.12: Operation and header files .

We now give an overview of the sender and receiver programs:

— qot_sender.h contains the declaration of the functions defined in qot_sender.c , as well as the definition of the structure qot_sender . This structure will be used to store the sender's oblivious key (called k in **Protocol 2** illustrated by Figure 2.16 in Chapter 2).

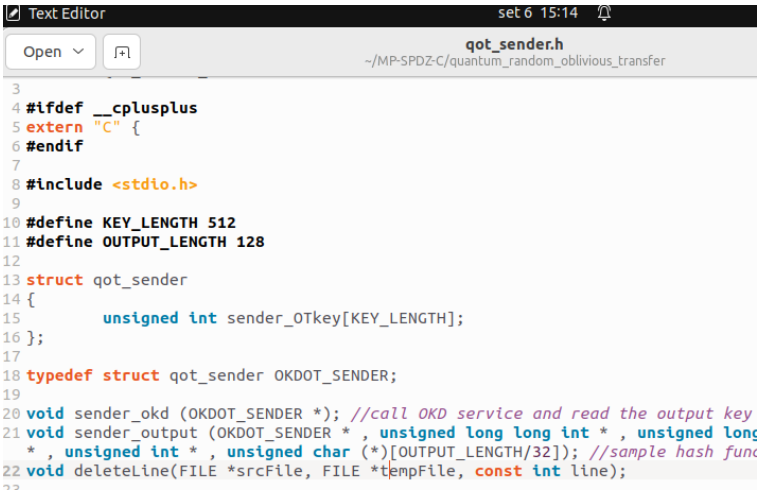


Figure 4.13: qot_sender.h

— qot_receiver.h contains the declaration of the functions defined in qot_receiver.c , as well as the definition of the structure qot_receiver . This structure will be used to store the receiver's oblivious key and auxiliary key (k_0 and x in Protocol 2), as well as the list of indexes where the auxiliary key is 0, and the list of indexes where it is 1 (I_0 and I_1 in Protocol 2).

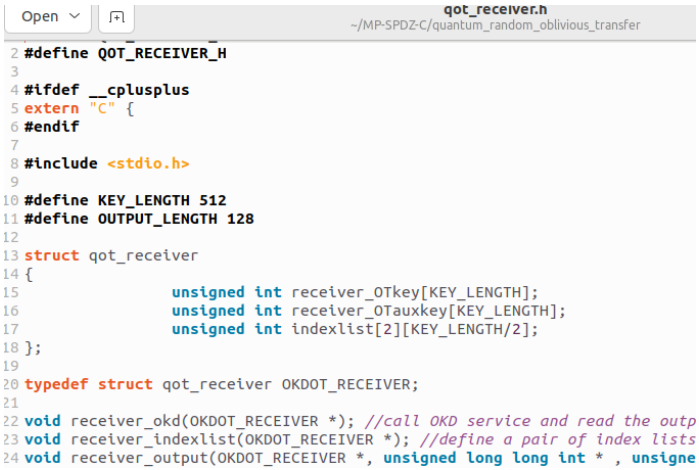


Figure 4.14: qot_receiver.h

— qot_sender.c defines two functions: sender_okd reads the sender oblivious key from a file and stores it in the sender structure; sender_output uses two arrays of random numbers, v_0 and v_1, to hash I_b and$I_{b\oplus 1}$. (see Step 3 of Protocol 2).

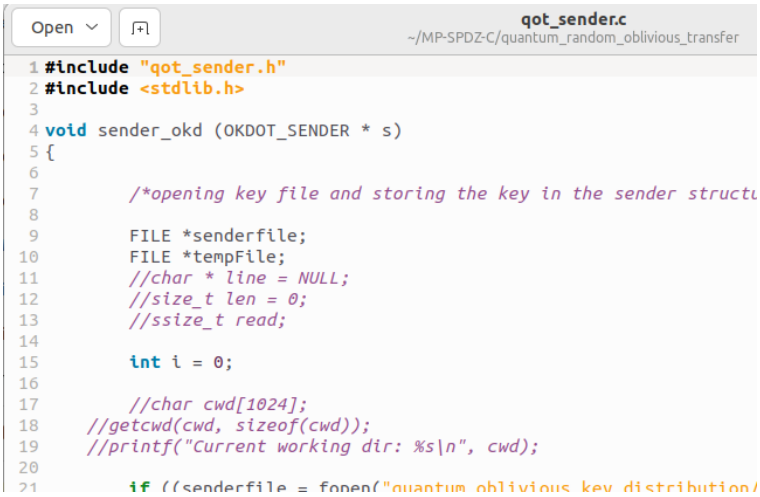


Figure 4.15: qot_sender.c

— qot_receiver.c defines three functions: receiver_okd reads the receiver oblivious key and auxiliary key from a file and stores them in the receiver structure; receiver_indexlist stores the indexes where the auxiliary key is 0 and the indexes where the auxiliary key is 1 in a two-dimensional array in the receiver structure (see Step 2 of Protocol 2); receiver_output uses an array of random numbers, v_b, to hash I_0 (see Step 4 of Protocol 2).

Figure 4.16: qot_receiver.c

→ This version of the random OT implementation does not yet receive the keys directly from the distribution system. Instead, the keys are read from files that we had to create and which are then stored in the `quantum_random_oblivious_transfer` directory.

Here in Figure 4.17 are the keys for Alice and Bob, which must be in this specific format.

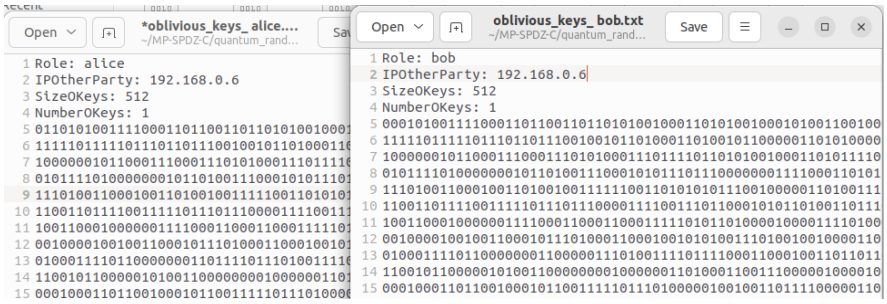


Figure 4.17: oblivious_keys_alice.txt and oblivious_keys_bob.txt

In order to integrate this implementation in the existing Yao software, the method `BaseOT::exec_base` defined in `BaseOT.cpp` was rewritten such that it calls our implementation of (Protocol 2) and uses it as a base OT. The generation of the random arrays v_0 and v_1 used to calculate hash functions is done inside the `BaseOT::exec_base` method, as well as the communications between the two players. The Makefile in MP-SPDZ must also be changed in order to link the library `libqokdot.a` .

4.4.1 Integration steps

[1.] First, We clone the git repository in [MP-SPDZ].

[2.] Before compiling the software, we need to install the following requirements : `automake build-essential git libboost-dev libboost-thread-dev libsodium-dev libssl-dev libtool m4 python texinfo yasm` .

[3.] Then, We put the quantum_random_oblivious_transfer folder in the MP-SPDZ directory. We Execute the make comand inside the folder quantum_random_oblivious_transfer to generate the `liboqokdot.a` library as shown in figure 4.18.

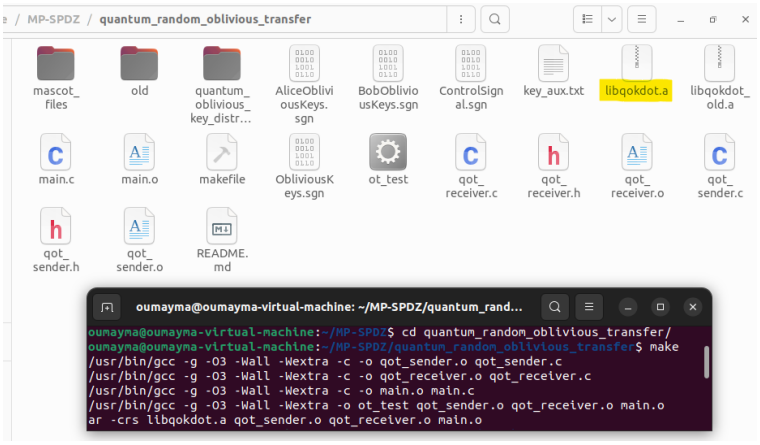


Figure 4.18: Generating `liboqokdot.a` library

[4.] We substitute the files :

MP-SPDZ/Makefile and

MP-SPDZ/OT/BaseOT.cpp

with the files :

quantum_random_oblivious_transfer/mascot_files/Makefile and

quantum_random_oblivious_transfer/mascot_files/BaseOT.cpp .

[5.] We check that the location of `liboqokdot.a` is correctly written in the Makefile in MP_SPDZ. Check that the location of `qot_receiver.h` and `qot_sender.h` is correctly written in `MP_SPDZ/OT/BaseOT.cpp`.

[6.] Finally, we need to change the name of the `oblivious_key.txt` files according to whether it is Alice or Bob to avoid confusion during compilation

4.4.2 Compiling the machine running the Yao protocol

make -j8 yao-party.x

→ This will also compile the files in `quantum_random_oblivious_transfer` and generate the library `libqokdot.a`. Or we can run the `make` command in the `quantum_random_oblivious_transfer` directory to compile these files only. We also check that the QOT module is properly functioning by running the QOT module executable and it runs successfully as shown in the figure 4.20 .

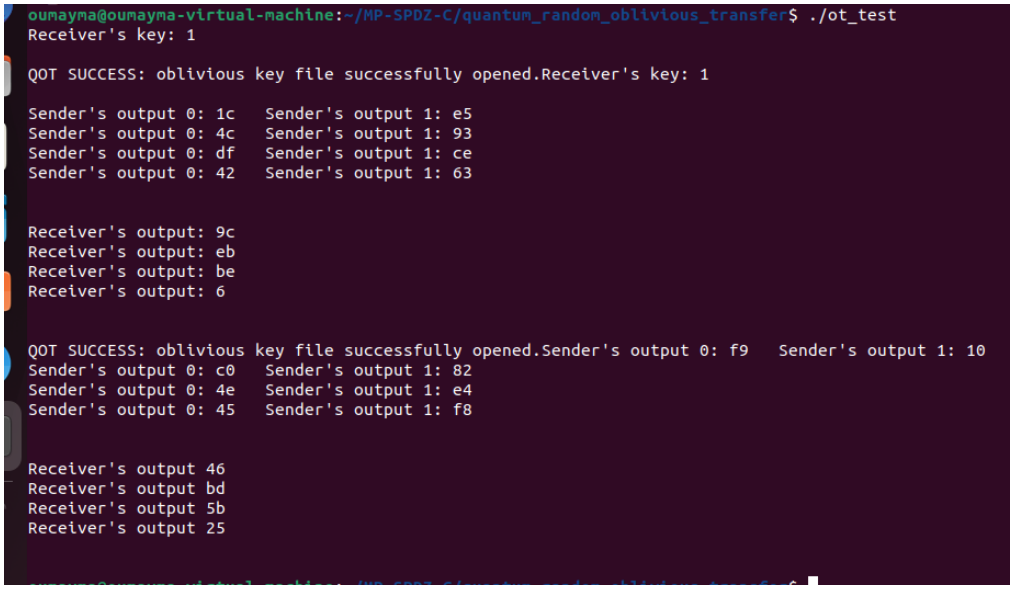


Figure 4.19: Testing QOT module

Then we compile our program with the function to be calculated as we did before :

```
./compile.py -B 64 service
```

To run the protocol with two parties on one machine, we run in seprate terminals :

•Garbler: `./yao-party.x -I -p 0 service`

•Evaluator: `./yao-party.x -I -p 1 service`

4.5 CARLA Driving Simulator

CARLA [27] is an open-source autonomous driving simulator. It was built as a modular and flexible API to address a range of tasks related to the autonomous driving problem. And it is grounded on Unreal Engine to run the simulation and uses the OpenDRIVE standard (1.4 as today) to define roads and urban settings.

→ We are using this simulator to create a kind of demo showing the expectations and a realistic view of our future system using the digital assets (city maps, buildings, vehicles),the flexible specification of sensor, environmental conditions, the full control of all static and dynamic actors and the map generation provided in CARLA to best reflect the desired system functioning image .

what CARLA can achieve ?

— Traffic manager ;

— Sensors from cameras to radars, lidar and many more ;

— Recorder to reenact a simulation step by step for every actor ;

— Open assets by offering different maps for urban settings with control over weather conditions and a blueprint library with a wide set of actors to be used ;

— ROS bridge and Autoware implementation for the integration of the simulator within other learning environments.

4.5.1 CARLA Architecture

The CARLA simulator consists of a scalable client-server architecture.

The server is responsible for everything related with the simulation itself: sensor rendering, computation of physics, updates on the world-state and its actors and much more.

The client consists of a sum of client modules controlling the logic of actors on scene and setting world conditions.

→ This is achieved by leveraging the CARLA API (in Python or C++), a layer that mediates between server and client that is constantly evolving to provide new functionalities.

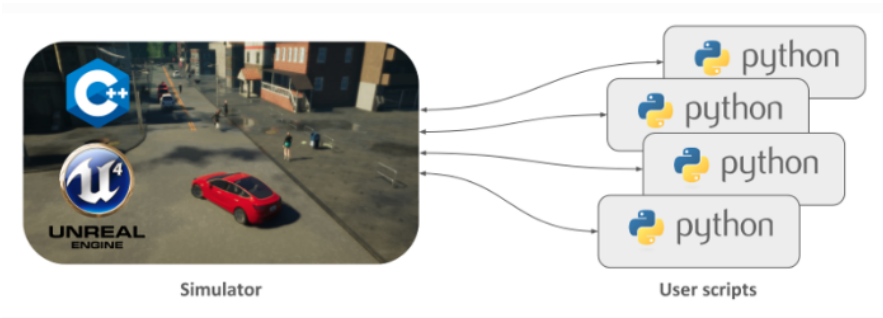


Figure 4.20: CARLA Simulator.

→ Control over the simulation is granted through an API handled in Python and C++ .

4.5.2 CARLA Requirements

The following requirements should be fulfilled before installing CARLA:

— **System requirements**. CARLA is built for Windows and Linux systems.

— **An adequate GPU**. CARLA aims for realistic simulations, so the server needs at least a 6 GB GPU. It is recommended 8 GB.

— **Disk space**. CARLA will use about 20 GB of space.

— **Python**. Python is the main scripting language in CARLA. CARLA supports Python 2.7 and Python 3 on Linux, and Python 3 on Windows.

— **Pip**. Some installation methods of the CARLA client library require pip or pip3 version 20.3 or higher.

— Two TCP ports and good internet connection. 2000 and 2001 by default. We need to make sure that these ports are not blocked by firewalls or any other applications.

→ Seeing these specifications, CARLA is resource intensive and requires a powerful machine to be able to run it. This is why we chose to install Carla on a dedicated machine available in the lab with all the necessary requirements.

4.5.3 First steps with CARLA

In this section, we will cover a standard workflow in CARLA, from launching the server and connecting the client, through to generating map, adding vehicles, sensors and generating some data to be used to determin vehicule location.

Launching CARLA and connecting the client

CARLA can be launched using the command line using the executable in the shell script in Linux and we launch CARLA from the command line:

```
cd /carla/root
```

```
./CarlaUE4.sh
```

To manipulate CARLA through the Python API, we need to connect the Python client to the server through an open port, the port can be chosen as any available port and is set to 2000 by default.

The client controls the simulator through the client and world objects. First we create a new script, then we add the following code to the start of the main function:

```
import carla
import random

# Connect to the client and retrieve the world object
client = carla.Client('localhost', 2000)
world = client.get_world()
```

Loading a map

In the CARLA API, the world object provides access to all elements of the simulation, including the map. Objects within the map: buildings, traffic lights, vehicles and pedestrians.

The CARLA server normally loads a default map (normally Town10) as shown in Figure 4.22.

We can launch CARLA with an alternate map, using the `config.py` script:

```
./config.py -map Town05
```

We can also use the world object to load a map from the client:

```
world.load_world('Town05')
```

Figure 4.21: CARLA Simulator -Town 10

Adding NPCs

Now we've loaded the map and the server is up and running we now need to populate our simulation with some vehicles, as shown in figure **??**, to simulate a real environment with traffic and other road users or non-player characters (NPCs).

→ To spawn vehicles, first, we need to select the vehicles we want from the blueprint library:

```
vehicle_blueprints = world.get_blueprint_library().filter('*vehicle*')
```

Now we have the blueprints, we need to find some appropriate spots in the map to spawn the vehicles. Each CARLA map provides pre-defined spawn points spread throughout the map on the roads for this purpose.

```
# Get the map's spawn points
spawn_points = world.get_map().get_spawn_points()

# Spawn 50 vehicles randomly distributed throughout the map
# for each spawn point, we choose a random vehicle from the blueprint library
for i in range(0,50):
    world.try_spawn_actor(random.choice(vehicle_blueprints, random.choice(spawn_points)))
```

Figure 4.22: CARLA Simulator -Spawning vehicles

→ we should also add a vehicle that will be the centerpoint of our simulation. we often refer to this vehicle as the "Ego vehicle" :

```
ego_vehicle = world.spawn_actor(random.choice(vehicle_blueprints),
random.choice(spawn_points))
```

Adding new props

Props are the assets that populate the scene, besides the map, and the vehicles. That includes streetlights, buildings, trees, and much more as shown in figure 4.24.

Figure 4.23: CARLA Simulator -Adding props

Spawning pedestrians

we want to spawn pedestrian in the simulation. This can be done at a random location using `world.get_random_location_from_navigation()`, or it can be chosen using coordinates gathered from the Unreal Editor.

An example of spawning pedestrian in Carla Simulation is shown in figure 4.24.

Figure 4.24: CARLA Simulator -Spawning pedestrians

→ Here we have described and presented some of the features, there are much more, but this will help us to make the demo to see how our future system function.

Conclusion

In this chapter we have discussed the technical choices that were made in terms of the technologies, libraries and tools used. Then we detailed the implementation steps of our service starting with the installation of the MP-SPDZ framework, the implementation of the yao protocol with our program and the integration of the quantum primitives. Finally we presented the CARLA simulation tool used to give a general overview of our solution.

General Conclusion

A survey of several research papers on VANET architecture, characteristics, applications, security, and security challenges are presented in this report in order to design and implement an SMC use case for Vanets . In this use case we aim to design an application for VANET drivers in order to indentify a service nearby the driver's location according to their preferences and without disclosing the privacy of their location.

To accomplish this, we used the concept of SMC by implementing the YAO'GC Protocol in our solution and also integrating quantum primitives to make the solution faster and secure even against a quantum computer attacks.

In fact, this project has given us a very interesting opportunity to learn about an innovative concept, namely VANETs, which are promising additions to our future intelligent transport systems, that have attracted the attention of automotive companies, academics and government agencies worldwide.

In terms of professional benefits, our time at the University of Aveiro, one of the most innovative universities in Portugal, allowed us to develop and strengthen skills useful for group and individual work, including the ability to: break down complex tasks into parts and steps,plan and manage time, develop stronger communication skills and refine understanding through discussion and explanation. But most importantly, conducting research and generating new knowledge is so beneficial for encouraging critical thinking and analytical skills through practical learning.

Our emergence in a high-tech professional and innovative environment allows us to imagine several perspectives on our work. We hope to see our application evolve into an integrated VANET platform with much more services, choices and functionalities for the user, as well as developing ergonomic interfaces in a user-friendly and innovative dashboard, making it easier to use and manipulate the application to ensure a pleasant and confortable journey for VANET drivers.

Finally, what we will remember from this experience is the method and organisation of a project, as we are used to working in a team, through this internship, we realized what a project methodology looks like, and what it implies (deadlines, follow-up, tests,presentations...).

Bibliography

[1] I. I. de Telecomunicações. « It - deti - university of aveiro ». [Accessed on 02/08/2022]. (2022), [Online]. Available: https://www.it.pt/AboutIT/Overview.

[2] U. of Aveiro. « Telecommunications institute - it ». [Accessed on 02/08/2022]. (2022), [Online]. Available: https://www.ua.pt/pt/deti/page/576.

[3] I. I. de Telecomunicações. « Organization ». [Accessed on 02/08/2022]. (2022), [Online]. Available: https://www.it.pt/AboutIT/Organization.

[4] Q. researchers team members. « Quests ». [Accessed on 03/08/2022]. (June 30rd, 2021), [Online]. Available: http://quests.av.it.pt/.

[5] M. K. Saggi and R. K. Sandhu. « A survey of vehicular ad hoc network on attacks security threats in vanets ». [Accessed on 03/08/2022]. (December 2014), [Online]. Available: https://www.researchgate.net/publication/295595335_A_Survey_of_Vehicular_Ad_Hoc_network_on_Attacks_Security_Threats_in_VANETs.

[6] N. A. et al. « A survey of vehicular ad hoc network on attacks security threats in vanets ». [Accessed on 06/08/2022]. (26 April 2022), [Online]. Available: https://www.hindawi.com/journals/wcmc/2022/6503299/.

[7] J. L. Muhammad Sameer Sheikh and W. Wang. « General architecture of vanet ». [Accessed on 13/08/2022]. (17 Aug 2019), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6720296/.

[8] D. T. RadhaKrishna Karne. « Review on vanet architecture and applications ». [Accessed on 11/08/2022]. (No.04 (2021),), [Online]. Available: https://www.researchgate.net/publication/295595335_A_Survey_of_Vehicular_Ad_Hoc_network_on_Attacks_Security_Threats_in_VANETs.

[9] M. S. Khan. « A survey of security services, attacks, and applications for vehicular ad hoc networks (vanets) ». [Accessed on 13/08/2022]. (Oct 2016,), [Online]. Available: https://www.researchgate.net/figure/General-architecture-of-VANET_fig1_311611240.

[10] Fireblocks. « What is mpc (multi-party computation) ». [Accessed on 15/08/2022]. (2022), [Online]. Available: https://www.fireblocks.com/what-is-mpc/.

[11] A. C. Yao. « Protocols for secure computations. in 23rd annual symposium on foundations of computer science(sfcs 1982), pages 160–164. publisher: ieee,1982 ». [Accessed on 15/08/2022]. (3-05 November 1982), [Online]. Available: https://ieeexplore.ieee.org/document/4568388.

[12] S. Latyšov. « A new standard: redesigning autonomous vehicle communication using multi-party computation ». [Accessed on 15/08/2022]. (2021-07-02), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6720296/.

[13] Y. Lindell. « Secure multiparty computation ». [Accessed on 17/08/2022]. (January 2021), [Online]. Available: https://cacm.acm.org/magazines/2021/1/249459-secure-multiparty-computation/fulltext.

[14] T. Chou and C. Orlandi. « The simplest protocol for oblivious transfer ». [Accessed on 18/08/2022]. (November 2021), [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3503045.

[15] R. Yackel. « What is homomorphic encryption ». [Accessed on 19/08/2022]. (July 6,2021), [Online]. Available: https://www.keyfactor.com/blog/what-is-homomorphic-encryption/.

[16] M. Keller. « MP-SPDZ: a versatile framework for multi-party computation ». [Accessed on 21/08/2022]. (2020), [Online]. Available: https://github.com/data61/MP-SPDZ/#readme.

[17] D. Rotaru. « How to choose your mpc framework? » [Accessed on 21/08/2022]. (30/04/2020), [Online]. Available: https://www.esat.kuleuven.be/cosic/blog/how-to-choose-your-mpc-framework/.

[18] Y. Lindell and B. Pinkas. « A proof of security of yao's protocol for two-party computation ». [Accessed on 24/08/2022]. (June 26, 2006), [Online]. Available: https://eprint.iacr.org/2004/175.

[19] K. Gkikas. « Yao's garbled circuit ». [Accessed on 23/08/2022]. (October 23, 2014), [Online]. Available: https://homepages.cwi.nl/~schaffne/courses/crypto/2014/presentations/Kostis_Yao.

[20] N. Yadav. « Scribe for lecture 9 1 yao's millionaires' problem 2 multi-party computation (mpc) problem ». [Accessed on 23/08/2022]. (16 Sept 2019), [Online]. Available: https://www.csa.iisc.ac.in/~cris/resources/e0_235/scribes/Lecture9.pdf.

[21] Bellare–Micali. « Oblivious transfer and applications. in: proceedings on advances in cryptology, crypto -89, pp. 547–557. » [Accessed on 04/09/2022]. (Springer-Verlag, Berlin (1989)), [Online]. Available: https://dl.acm.org/doi/abs/10.5555/88314.

[22] S. Pohlig and M. Hellman. « An improved algorithm for computing logarithms overgf(p)and its cryptographic significance (corresp.) » [Accessed on 04/09/2022]. (January 1978), [Online]. Available: https://ieeexplore.ieee.org/document/1055817.

[23] D. Adrian, K. Bhargavan, Z. Durumeric, *et al.* « Imperfect forward secrecy: how diffie-hellman fails in practice ». [Accessed on 04/09/2022]. (12 October 2015), [Online]. Available: https://dl.acm.org/doi/10.1145/2810103.2813707.

[24] M. B.Santos, A. N.Pinto, and P. Mateus. « Quantum and classical oblivious transfer: a comparative analysis ». [Accessed on 04/09/2022]. (21 May 2021), [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/qtc2.12010.

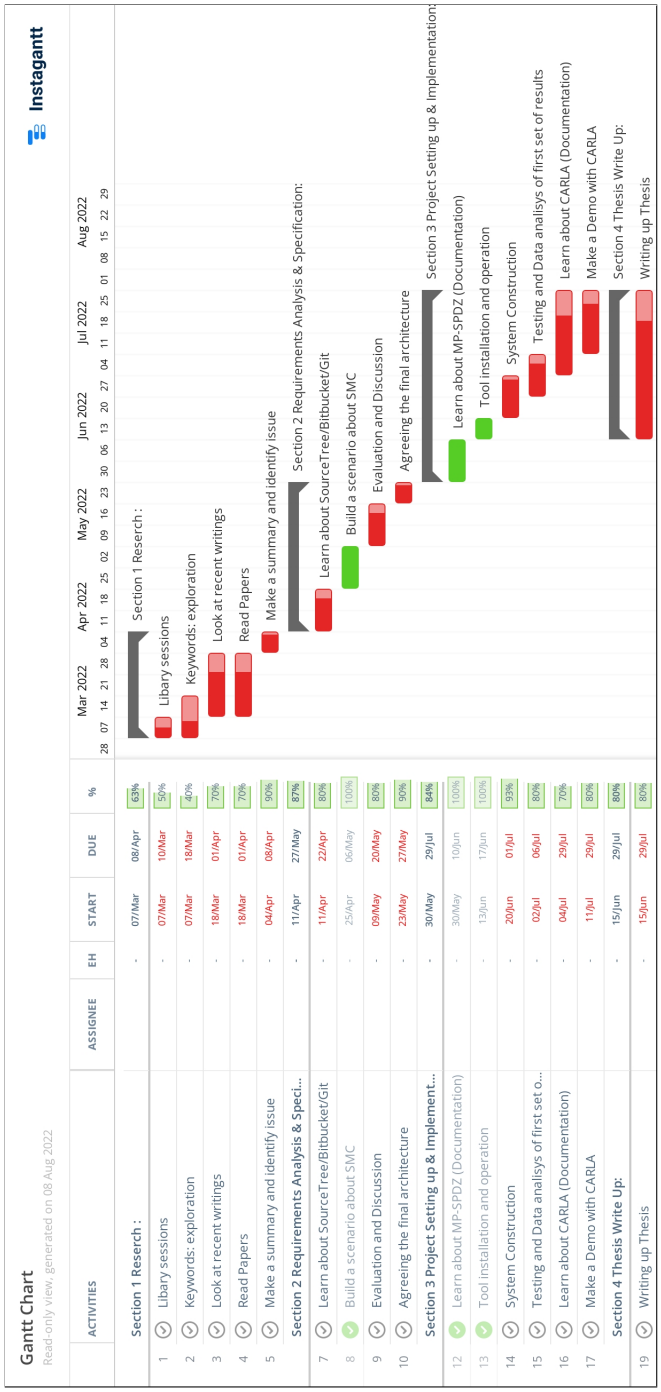
[25] M. Keller. « Mp-spdz documentation ». [Accessed on 31/08/2022]. (2020), [Online]. Available: https://mp-spdz.readthedocs.io/_/downloads/en/latest/pdf/.

[26] M. Keller. « Welcome to mp-spdz's documentation! » [Accessed on 01/09/2022]. (2021), [Online]. Available: https://mp-spdz.readthedocs.io/en/latest/.

[27] M. Pawelczyk, S. Bielawski, J. van den Heuvel, T. Richter, and G. Kasneci. « Carla documentation ». [Accessed on 07/09/2022]. (2021), [Online]. Available: https://carla.readthedocs.io/en/latest/.

Appendix

Appendix 1. Gantt Chart



ملخّص

الهدف الرئيسي من هذا التقرير هو عرض تصميم و تطوير لبرنامج لخدمة سائقي فانيت (VANET) وهي التكنولوجيا التي تستخدم سيارات متحركة في الشوارع بحيث يتم إنشاء اتصال إنترنت عبر هذه الشبكة المتنقلة . تهدف هذه الخدمة لتسهيل عملية البحث عن مكان او خدمة حسب الحاجة وذلك باستخدام تكنولوجيا الحوسبة الآمنة متعددة الأطراف بما يعرف ب (SMC) . الخاصية المميزة لهذا لهذا التطبيق هي أنه يحافظ على سرية بيانات السائق من خلال استخدام بروتوكول (YAO'GC) وتكنولوجيا الحواسب الكمية. وفيما يلي بيان تفصيلي لهذا التقرير: أولا عرض عام لسياق المشروع، ثم تفسير جميع المفاهيم والتكنولوجيات الجديدة المستعملة. ثالثاً، تحليل الاحتياجات والدراسة المفهومية، وأخيراً الجزء المتعلق بالتصميم.

كلمات مفاتيح : VANET، SMC، Yao، GC، الحاسوب الكمي

Résumé

L'objectif principal de ce projet de fin d'étude est de présenter et de développer un cas d'utilisation pour Les calculs multi-parties sécurisés (MPC) pour les conducteurs VANET en offrant un service qui permet d'identifier facilement un service à proximité selon leurs préférences.Ce projet a été développé dans le cadre d'un stage de PFE au département IT de l'Université d'Aveiro (Portugal). la répartition de ce rapport est comme suit : d'abord une présentation générale du contexte du projet, ensuite, l'état de l'art dans lequel tous les nouveaux concepts et technologies sont définis. Troisièmement, l'analyse du besoin et l'étude conceptuelle, et enfin la partie réalisation. La particularité de cette application est qu'elle préserve la confidentialité des données du conducteur grâce à l'utilisation du protocole Yao'GC et de la technologie quantique.

Mots clés : VANET, MPC, Yao Protocole, GC, primitives Quantiques.

Abstract

The main objective of this thesis is to present and develop an SMC use case for VANET drivers by offering an infotainment service that allows VANET drivers to easily find a nearby service according to their preferences. This project was developed in the context of a final year internship at the IT department of the University of Aveiro (Portugal).It includes first a general presentation of the project context, secondly, the state of the art in which all the new concepts and technology are defined. Thirdly analysis of the needs and the conceptual study, finally the implementation part of the application .

The particularity of this application is that it preserves the confidentiality of the driver's data through the use of the Yao'GC protocol and the technology of quantum primitives.

Keywords : VANET, SMC, Yao Protocol, GC, Quantum primitives.

جامعة سانتياغو ـ افيرو ـالبرتغال الهاتف : 200 370 234 351+ الفاكس : 089 370 234 351+ البريد الالكتروني : geral@ua.pt
Campus Universitário de Santiago-Aveiro,Portugal **Tél** : +351 234 370 200 **Fax** : +351 234 370 089 **Email** : geral@ua.pt

2 نهج أبو الريحان الباروني 2080 أريانة الهاتف : 164 706 71 الفاكس : 698 706 71 البريد الالكتروني : isi@isim.rnu.tn
2, Abou Raihane Bayrouni 2080 l'Ariana **Tél** : 71 706 164 **Fax** : 71 706 698 **Email** : isi@isim.rnu.tn