UNIVERSITÀ DEGLI STUDI DI PALERMO

Dottorato di Ricerca in INGEGNERIA DELL'INNOVAZIONE TECNOLOGICA
Dipartimento di Ingegneria
SSD: ING-INF/05

EFFICIENT AND SECURE ALGORITHMS FOR MOBILE CROWDSENSING THROUGH PERSONAL SMART DEVICES

IL DOTTORE
Ing. Federico Concone

IL COORDINATORE
Ch.mo Prof. Salvatore Gaglio

IL TUTOR
Ch.mo Prof. Giuseppe Lo Re

IL CO-TUTOR
Prof. Sajal K. Das
Missouri University of Science and Technology

CICLO XXXIII
ANNO CONSEGUIMENTO TITOLO: 2021

To my family and to my love.

Thanks for your great support and continuous care.

Abstract

The success of the modern pervasive sensing strategies, such as the Social Sensing, strongly depends on the diffusion of smart mobile devices. Smartwatches, smartphones, and tablets are devices capable of capturing and analyzing data about the user's context, and can be exploited to infer high-level knowledge about the user himself, and/or the surrounding environment. In this sense, one of the most relevant applications of the Social Sensing paradigm concerns distributed Human Activity Recognition (HAR) in scenarios ranging from health care to urban mobility management, ambient intelligence, and assisted living.

Even though some simple HAR techniques can be directly implemented on mobile devices, in some cases, such as when complex activities need to be analyzed timely, users' smart devices should be able to operate as part of a more complex architecture, paving the way to the definition of new distributed computing paradigms. The general idea behind these approaches is to move early analysis towards the edge of the network, while relying on other intermediate (fog) or remote (cloud) devices for computations of increasing complexity.

This logic represents the main core of the *fog computing* paradigm, and this thesis investigates its adoption in distributed sensing frameworks. Specifically, the conducted analysis focused on the design of a novel distributed HAR framework in which the heavy computation from the sensing layer is moved to intermediate devices and then to the cloud. Smart personal devices are used as processing units in order to guarantee real-time recognition, whereas the cloud is responsible for maintaining an overall, consistent view of the whole activity set. As compared to traditional cloud-based solutions, this choice allows to overcome processing and storage limitations of wearable devices while also reducing the overall bandwidth consumption. Then, the fog-based architecture allowed the design and definition of a novel HAR technique that combines three machine learning algorithms, namely k-means clustering, Support Vector Machines (SVMs), and Hidden Markov Models (HMMs), to recognize complex activities modeled as sequences of simple micro-activities.

The capability to distribute the computation over the different entities in the network, allowing the use of complex HAR algorithms, is definitely one of the most significant advantages provided by the fog architecture. However, because both of its intrinsic nature and high degree of modularity, the fog-based system is particularly prone to cyber security attacks that can be performed against every element of the infrastructure. This aspect plays a main role with respect to social sensing since the users' private data must be preserved from malicious purposes. Security issues are generally addressed by introducing cryptographic mechanisms that improve the system defenses against cyber attackers while, at the same time, causing an increase of the computational overhead for devices with limited resources. With the goal to find a trade-off between security and computation cost, the design and definition of a secure lightweight protocol for social-based applications are discussed and then integrated into the distributed framework. The protocol covers all tasks commonly required by a general fog-based crowdsensing application, making it applicable not only in a distributed HAR scenario, discussed as a case study, but also in other application contexts.

Experimental analysis aims to assess the performance of the solutions described so far. After highlighting the benefits the distributed HAR framework might bring in smart environments, an evaluation in terms of both recognition accuracy and complexity of data exchanged between network devices is conducted. Then, the effectiveness of the secure protocol is demonstrated by showing the low impact it causes on the total computational overhead. Moreover, a comparison with other state-of-art protocols is made to prove its effectiveness in terms of the provided security mechanisms.

Acknowledgments

I want to thank all the people who helped me along the way, both professionally and personally.

First, I owe a huge debt of gratitude to my tutor *Prof. Giuseppe Lo Re* who gave me the opportunity to conduct this research, for his constant encouragement and guidance in my academic effort. Your brilliant insights have been a tremendous help for my work.

I wish to express my gratitude to my co-tutor *Prof. Sajal K. Das* for providing guidance and feedback during my visiting period. The meetings and conversations were vital in inspiring me to think outside the box.

Special thanks to *Dr. Marco Morana* for being so incredibly supportive and for sharing with me his invaluable experience during our close collaboration. Without his dedication and hard work, none of this would have been possible. I sincerely appreciate the time you took to guide me and my work.

I would like to thank all *members of the NDS research group* for the several useful discussions and for being a great source of support. It is always a great pleasure and honor to work with all of you.

Last but not least, I can't begin to express how thankful I am to my *family*, all of you have supported me and had to put up with my stresses and moans for the past three years of study.

Finally, to *my love*, thank you for making me a better person and being my lifeline whenever I needed it. Thank you for coming into my life, you are simply amazing!

Contents

List of Figures

List of Tables

Glossary

AES	Advanced Encryption Standard
AI	Artificial Intelligence
CDC	Cloud Data Center
CR	Compression Ratio
DH	Diffie-Hellman
DoS	Denial of Service
ECC	Elliptic-Curve Cryptography
ED	Edge Device
FD	Fog Device
GPS	Global Positioning System
HAR	Human Activity Recognition
HMM	Hidden Markov Model
IoT	Internet of Things
JSON	JavaScript Object Notation
KFCV	K-Fold Cross Validation
KMS	Key Management Server
LOOCV	Leave-One-Out Cross Validation
MitM	Man in the Middle
ML	Machine Learning
OSN	Online Social Network
RSA	Rivest-Shamir-Adleman
SP	Saving Percentage
SVM	Support Vector Machine
TLS	Transport Layer Security

X3DHKA	eXtended Triple Diffie-Hellman Key Agreement
XML	eXtensible Markup Language

Chapter 1

Introduction

Nowadays, users are provided with an increasingly rich set of options for transmitting and sharing personal and non-personal information, thus generating massive amounts of data that may be used for a myriad of societal applications. These data represent the starting point for a new generation of services whose goal is to process users' context observations and to retrieve a high-level information of useful interest to the community and/or to user himself. For example, an intelligent system deployed on a university campus can take advantage of sensory data produced by users' smart devices to provide services to students and academics, such as the efficient management of energy resources or shared spaces inside the campus, i.e., parking areas, classrooms, school cafeterias and so on [Agate et al. (2018)].

This phenomenon is defined as *Social Sensing*, i.e. the act of collecting observations on the physical environment by human beings or devices acting on their behalf [Aggarwal and Abdelzaher (2013)]. The novel idea introduced by this paradigm is the concept of *user as a sensor*, where humans produce data while interacting with the surrounding environment, whether it is physical or virtual. Like the traditional sensors, they can sense the physical world and are embedded in spaces that are interesting to measure. Moreover, they extend these functionalities by integrating the ability of analyzing complex scenes, detecting anomalies, and prioritizing information transfer. Hence, it is clear that the study of Social

Sensing paradigm and its applicability to real-world applications is of paramount importance to better describe many human-centric eco-systems, ranging from *Ambient Intelligence* to *Health Care.*

Given the central role of the individuals, the success of Social Sensing-based systems largely depends on the manner in which users are involved in the sensing process. They may be *implicit* information sources or *explicit* information sources; the first case occurs, for example, when users use *smart personal devices* (smartphone, tablet, smartwatch) that enables the learning of the users' habits without providing explicit information. Conversely, users become explicit sources of information, for example, when sharing content through OSNs. Although both visions offered by Social Sensing were investigated during the PhD period, the study conducted here pays more attention to the concept of users as *implicit* information sources. In this sense, one of the most attractive scenarios is distributed Human Activity Recognition (HAR), in which several entities cooperate to infer user's physical activity by means of the raw data produced by the motion sensors embedded in their smart devices.

The recognition of human activities plays a very important role in the modern applications, as it enables more complex and complete services to be provided to the end users. For example, in a *Health Promoting* scenario, where a community of people who play sports in the same place, several devices can interact to motivate the users to achieve a certain result. The users' smart devices could recognize the activities performed by one or more users, and share related information (e.g., elapsed time, speed, calorie consumption, heart rate) with the community in order to stimulate the users in increasing their exercise, thus resulting in greater physical benefits. Scaling up from individuals to groups, that is, *group activity recognition*, enables the monitoring of other impressive scenarios such as the understanding of social behaviors. For example, monitoring the eating together activity can be an aspect of social health and well-being of people. Generally, getting informed of occurred group activities, and situations within group activities, may benefit both a tracker (i.e., a person, group or any consumer who is interested in receiving information about groups) and a trackee (i.e., a person or group being tracked who might also receive service/s from trackers) [Abkenar et al. (2019)].

This simple case study, as well as many other real examples, highlights how HAR could be of tremendous relevance in the Social Sensing context. For this reason, this thesis investigates the recognition of human activities for large-scale applications by means of motion sensors included in the smart devices used during the daily life. To support this challenge, during the PhD, the *participatory sensing* [Burke et al. (2006)] and *crowdsensing* [Ganti et al. (2011)] data collection paradigms were analyzed and adopted for achieving the application specific goal.

1.1 Motivations and Goals

The increasing necessity of more and more effective HAR algorithms has induced the research community to study and define different techniques over the years. For example, one of the most reliable solution is represented by the *Google's Activity Recognition* API that allows to detect activities by periodically processing sensor data using machine learning (ML) models. This solution, as well as many other in the literature, proves that the activity recognition can be performed on board the device, thus not requiring additional resources and/or entities. However, an early analysis about the HAR methods implemented in smart devices shows the existence of several limitations. For example, some approaches act as a black-box not providing neither a way to understand what mechanisms are behind it, nor intermediate information to supervise the running of the recognition process. Hence, a preliminary study addressed the definition and discussion of a sample HAR system with the aim to better investigate how difficult is recognizing activities on board the smart devices, as well as the performance achieved by such a system. Results suggested that, in the case of simple activities, a simple and well-known ML technique, i.e. k-nearest neighbors (K-NN), is sufficient to obtain very good performances in terms of recognition accuracy and algorithmic efficiency.

Nevertheless, the current market provides users with other devices that are more resource constrained compared to smartphones, e.g. smartwatches and other wrist-worn devices. In this sense, inferring the activity could be a very intensive task, especially when the data processing may negatively impact the battery life. Google itself in the Activity Recognition APIs web page states: “To optimize resources, the API may stop activity reporting if the device has been still for a while, and uses low power sensors to resume reporting when it detects movement" [Google]. This suggests that the adopted technique may negatively impact on the resource consumption of the device in which the algorithm is running.

To alleviate this issue, cloud computing could provide a feasible solution to move heavy computation towards the cloud while using the mobile device as a pure sensing platform. However, all the benefits brought by this approach could be negligible in real-time applications where data are continuously transferred from/to the cloud, thus causing a large amount of network overhead.

With the aim to bypass these limitations, the fog computing paradigm [Bonomi et al. (2012)] is investigated and applied to the proposed system. In 2012, this paradigm was introduced by Cisco as an extension of the cloud computing at the edge of the network. Today, the fog has been widely accepted as a reasonable alternative to the cloud when dealing with large amounts of data that need to be processed locally and timely. Thus, due to its intrinsic pervasive nature, HAR represents the ideal scenario where fog computing can provide a significant improvement in system performance. In particular, the general idea, which supports this doctoral thesis, is that data are processed as close as possible to each user, i.e. users' smartwatches and smartphones, so as to guarantee real-time recognition, whereas a remote cloud infrastructure is responsible for maintaining an overall, consistent, view of the whole activity set.

The adoption of the fog computing, for sure, offers tremendous benefits in designing a distributed HAR system, but introduces a number of issues related to the security of the data acquisition and processing algorithms, as well as data exchange and storage. In these applications the cyber attackers, whether they are real or artificial entities, aim to worsen the performance of the system under different aspects. Confidentiality, availability, and integrity can be achieved by adopting and setting in a proper way the existing cryptographic schemes. Other issues, such as the user privacy, are dependent on the specific application domain and require the designing of ad-hoc protocols. For example, in a HAR scenario, fitness data collected by wearable devices can include heart rate, location, calories consumption, stress level, and other data that can reveal the user's identity, ethnicity, disease risks, and other sensitive information [Lyu et al. (2017)]. Users are also concerned that their data might be sold to third parties without their consent. This leads to two requirements that have been addressed in the proposed system: the HAR algorithm must recognize the activities without associating them to a specific user; data exchanged between entities in the fog architecture must be secured to avoid any attackers to steal sensory data that may contain private information.

For this reason, lightweight security protocols for mobile crowdsensing are required. In the scenario considered here, this point has been addressed and resulted in the definition of an innovative and general Secure Mobile Crowdsensing Protocol (SMCP). Specifically, it enables the protection both data shared in the Social Sensing application and the user privacy, without affecting the performances of the underlying system in terms of computation overhead. The generality and effectiveness of both SMCP and the HAR framework are proved by an exhaustive experimental analysis. The HAR framework is evaluated in terms of accuracy of the recognition process, while the performances of the SMCP have been evaluated in terms of computation time required for completing all cryptographic tasks that implement the secure protocol. Moreover, by using the HAR framework as case study, a comparative analysis between SMCP and two state-of-the-art schemes has been presented. Results show that SMCP allows to achieve a higher degree of security while maintaining a low computational cost.

1.2 Contributions

The main contributions of the doctoral work presented in this dissertation are:

- The definition of a fog architecture for recognizing complex human activities, in which different devices cooperate to understand the users' behavior. Data are processed as close as possible to each user. In the scenario considered here, the processing units are the users' smartwatches and smartphones, so as to guarantee real-time recognition, whereas a remote cloud infrastructure is responsible for maintaining an overall, consistent view of the whole activity set. By adopting such a general architecture in different application scenarios, the output provided by a single fog device could be merged with those coming from the users' community to enable more advanced services. Elderly people living in a nursing home could be monitored by means of unobtrusive wrist-worn devices (one per user), while a few smartphones (or any other device) owned by the home could be used to perform the activity recognition. In such a scenario, data from every user can be processed at the cloud level to define a global normal behavior that can be exploited to reveal warning or dangerous situations. In wider terms, data coming from the community could be used to support the recognition process itself. If a number of people visit a certain location, GPS data from multiple users could reveal the relationships between an activity and the place where it is performed. This could improve the system performance by limiting the recognition process to some of the most likely activities.

- A novel HAR technique that combines three machine learning algorithms - k-means clustering, support vector machines (SVMs), and hidden Markov models (HMMs) - to recognize complex activities modeled as sequences of simple micro-activities.
- The definition of a novel, lightweight, secure message exchange protocol that covers all the tasks commonly required by a general fog-based crowdsensing application.
- The adoption and performance evaluation of the secure protocol on a real case study in which edge/fog devices are exploited to perform the human activity recognition.
- A comparative analysis of SMCP and two state-of-the-art security protocols, whose results show that SMCP allows to achieve a higher degree of security while maintaining a low computational cost.

1.3 Dissertation Outline

Within the previous sections of this chapter, the background and the motivations for this doctoral thesis were described. Specifically, the Social Sensing paradigm and the concept of user as an implicit source of information were introduced; thus, highlighting the challenges it brings, as well as the opportunities it creates. In the light of such challenges, the contributions of this thesis were provided.

The remainder of the dissertation follows a bottom-up approach to describe the proposed framework.

In Chapter 2, the distributed fog infrastructure, as well as some application scenarios and the addressed case study, are described; then, the algorithm for recognizing complex human activities, together with an exhaustive description of each phase involved in the recognition process, are investigated and discussed.

In Chapter 3, the security requirements of the fog-based architecture in the context of the distributed HAR system are discussed. This is followed by a detailed description of the preliminary secure enrollment phases provided by the secure protocol.

In Chapter 4, the experimental setup and the relative results for both HAR and secure protocol are discussed. Firstly, the HAR process is evaluated in terms of detection performance and its impact on the infrastructure. Then, the secure protocol is analyzed by means the message preparation time and size, while a comparative analysis with two other security protocols is discussed at the end of this chapter.

Finally, Chapter 5 concludes this doctoral thesis by summarizing the obtained results, as well as the future research directions.

1.4 Publications

Parts of the work in this thesis have been published in several referred conference proceedings and journals:

- **F. Concone**, G. Lo Re, and M. Morana. SMCP: a Secure Mobile Crowdsensing Protocol for fog-based applications. Human-centric Computing and Information Sciences, 10, 2020.

- **F. Concone**, G. Lo Re, and M. Morana. A Fog-Based Application for Human Activity Recognition Using Personal Smart Devices. ACM Transaction Internet Technology, 19(2), March 2019.

- A. Pratap, **F. Concone**, V.S.S. Nadendla, and S.K. Das. Three-dimensional matching based resource provisioning for the design of low-latency heterogeneous IoT networks. In Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWIM '19, page 79-86, New York, NY, USA, 2019. Association for Computing Machinery.

- V. Agate, **F. Concone**, and P. Ferraro. WiP: Smart Services for an Augmented Campus. In Proceedings of the 4th IEEE International Conference on Smart Computing (SMARTCOMP 2018), Taormina, Italy, June 2018.
- **F. Concone**, P. Ferraro, and G. Lo Re. Towards a Smart Campus Through Participatory Sensing. In Proceedings of the 4th IEEE International Workshopon Sensors and Smart Cities (SSC 2018), pages 393-398. IEEE, 2018.
- **F. Concone**, S. Gaglio, G. Lo Re, and M. Morana. Smartphone Data Analysis for Human Activity Recognition. In Floriana Esposito, Roberto Basili, Stefano Ferilli, and Francesca A. Lisi, editors, AI*IA 2017 Advances in Artificial Intelligence, pages 58-71, Cham, 2017. Springer International Publishing.

Chapter 2

Distributed Human Activity Recognition

Sensor-based HAR has been widely addressed in the literature and most of the proposed solutions use a single mobile device to perform data collection and simple activity recognition [Li et al. (2015); Mannini et al. (2017)]. Unfortunately, in order to perform more intensive tasks, e.g., real-time classification of *complex activities*, mobile devices with limited resources need to be supported by a solid infrastructure to capture, manage, process, and store data coming from heterogeneous sensors.

Novel distributed computing paradigms have been proposed to develop pervasive systems in which a number of different devices can easily cooperate, proportionally to their computational capabilities and power requirements, in the accomplishment of complex tasks. The earliest solution, known as *cloud* computing, relied on a remote computing and storage infrastructure aimed at providing services according to predefined models, such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The limitation of this approach has quickly become clear since data captured by the *edge* of the network needed to be continuously transferred to the *cloud* in order to be processed, making this solution not suitable for real-time applications. As a consequence, *edge* computing and *fog* computing paradigms have been proposed with the aim of moving part of the computations towards the boundary of the network. A relevant survey of edge and fog computing architectures, applications, and research issues in the context of IoT is presented in [Omoniwa et al. (2019)].

This part of the thesis focuses on the design of a fog-based architecture, whose edge layer consists of cheap wearable devices with limited computational resources, e.g., wrist-worn smart devices. Such an architecture can be adopted to support a number of applications; for instance, wrist-worn devices could be efficiently used to obtain data about the environmental conditions, the weather, or the urban mobility/traffic congestion in a certain area. Nevertheless, they are not particularly suitable for performing intensive computations, nor for continuously transferring data over the network. These tasks, instead, can be properly accomplished by other devices that are close to the users, e.g., their smartphones or laptop computers. These (*fog*) units are characterized by a greater computing power and are usually provided with network interfaces that allow them to communicate in real-time with remote (*cloud*) data centers.

2.1 Existing Approaches for HAR

In recent years, Human Activity Recognition has become a relevant research area due to its suitability for different application scenarios.

The recognition of human activities has been generally approached focusing on vision or sensor-based solutions. In the first case, video sequences that capture the user's movements and gestures are analyzed. These techniques present some issues [Lara and Labrador (2013)] that limit their implementation in many real-world scenarios. The first is that video processing techniques are computationally expensive, thus they can be rarely executed in real-time on resource constrained devices. Moreover, the performance of these systems is strictly dependent on the position of the camera and the appearance of the scene, and therefore the recognition is often limited to indoor environments.

To overcome these limitations, many HAR techniques exploiting sensors directly carried by the users have been presented in the literature. Early solutions were based on acceleration sensors only [Ravi et al. (2005)]; however, since a single sensor is not suitable to describe very complex activities, several works proposed to merge information provided by multiple sensors. For example in [Bao and Intille (2004)], the authors present a system that acquires data from five biaxial accelerometers, worn simultaneously on different parts of the body, to recognize both simple and complex activities. The system presented in [Lester et al. (2006)] combines heterogeneous sensors, e.g., accelerometers and gyroscopes, microphones, GPS, in order to improve the recognition performances. Unfortunately, approaches based on wearable sensors are not suitable for real application scenarios due to their intrusiveness [Patel et al. (2012)].

Recent HAR techniques exploit the widespread diffusion of smart devices. In [Cvetković et al. (2016)] the authors present a system that aims to improve the quality of life of diabetic patients combining machine learning and symbolic reasoning techniques. Smartphone sensors are used to recognize some activities in order to trace patients' fatigue while performing their daily routines. In [Kwon et al. (2014)], the authors describe an unsupervised learning approach to recognize human activities using smartphone sensors. The recognition process is strictly dependent on the number of clusters chosen during the design phase, and thus distinct activities could be erroneously merged into one, or different instances of the same activity could be seen as unrelated. One of the best-performing HAR frameworks is proposed by Google [Google (2016)], as it is API level 1. However, these APIs represent a black box, and the developers are not able to use intermediate results as part of their systems, nor to provide any feedback to the activity recognition routine. For this reason, the Google framework can be only used to develop some simple Android applications or as a reference for comparing novel activity recognition techniques. In [Concone et al. (2017)], a framework based on smartphone embedded accelerometer and gyroscope sensors for real-time simple activity recognition is presented.

More recently the focus has moved to the recognition of more complex activities that can be modeled as a composition of simple actions. The authors of [Ryoo and Aggarwal (2009)] propose a description-based approach that allows to encode a complex activity through a context-free grammar (CFG), and to model it as an interaction between simpler activities. Similar approaches are used in video-based activity recognition, where a set of silhouettes can be extracted and analyzed to describe a particular human activity. For example in [Ogale et al. (2007)], a probabilistic context-free grammar (PCFG) is built from atomic actions. In [Gaglio et al. (2015)], a Kinect device is used to observe the user, and each activity is modeled as a spatio-temporal evolution of known postures extracted by some joints of the human body.

Some other works exploit probability based-algorithms, such as Conditional Random Fields (CRFs) [Lafferty et al. (2001)] and Hidden Markov Models (HMMs) [Rabiner and Juang (1986)], to model complex activities. In [Li et al. (2015)], a framework based on an adaptive HMMs is presented. Each complex activity is modeled as a sequence of simple activities performed by the user, and user's personal experience is considered as *a priori* information to train HMMs. In addition, unlike conventional methods that consider all data from sensors in computational process, such system proposes an adaptive Viterbi algorithm to speed up the classification.

Lately, the ever-increasing need for measuring large-scale phenomena has encouraged new approaches that aim at analyzing data captured from different entities. The Mobile CrowdSensing System (MCS) [Cardone et al. (2016, 2013)] provides some activity recognition and geofencing algorithms that are optimized to meet computational and power constraints of smartphone devices. In particular, the activity recognition sub-system allows to detect three kinds of activities (*walking*, *running*, and *phone still*), whilst geofencing aims to find and delimit the geographic area where a certain activity, or event, occurs.

To obtain scalable and time-efficient solutions, several works exploit the cloud computing paradigm to provide HAR services according to the most common delivery models, such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Unfortunately, cloud data centers are usually far away from the end devices/users [Mell and Grance (2011)], making the development of real-time applications quite critical. Some works focused on combining cloud computing and mobile devices taking the best of both worlds. In [Chun et al. (2011)], a system that allows to run mobile applications on the cloud is described. The basic idea of such work consists in creating and migrating an image of the smartphone to the cloud to perform CPU-intensive tasks on servers that have more resources than a mobile device. Another example is presented in [Zhang et al. (2011)], where a model that permits to decompose a mobile application in several components is proposed. Each component can be run either on a mobile device or migrated to the cloud, so as to overcome any computation or storage constraints.

As mentioned previously, the fog paradigm has been recently adopted in several application scenarios [Dastjerdi and Buyya (2016); Perera et al. (2017)] to move data processing close to the point where data are produced, such as by performing resource-expensive computing in lightweight servers placed at the edge of the network. A common scenario addressed by fog computing is the distributed video surveillance, in which traditional client-server architectures would not allow to transmit and analyze huge amounts of video streams efficiently [Hong et al. (2013)].

Just a few applications of fog computing in a HAR scenario have been presented in the literature. CARDAP [Jayaraman et al. (2014)] is a fog-based data analytics platform for supporting MCS applications in a smart city. The main goal of this system is to perform real-time recognition of the citizens' activities by analyzing data collected by mobile and Internet of Things devices. In [Perera et al. (2015)] is described a general platform addressing three different scenarios (i.e., environmental monitoring, rehabilitation, health) in which wearable sensors are used to measure air quality, user's movements, and sounds. In such a framework, Internet connected objects (ICOs) are used at the edge of the network, while user smartphones are exploited as intermediate gateways. Wearable sensors are also used in FAAL [Vora et al. (2017)], a fog-based patient monitoring system that traces the user's movements to recognize neurological diseases. Finally in [Cao et al. (2015)], a fog-based platform designed to detect user's falls in an e-health scenario is described. This system distributes the fall detection task among edge devices and the cloud, allowing lower response time and energy consumption than traditional non-fog approaches.

2.2 Algorithms for Human Activity Recognition

The human activities can be intuitively considered as sequences of recurrent patterns in raw data captured from smartphone's sensors. Many HAR algorithms have been presented in the literature, although their application is frequently re-

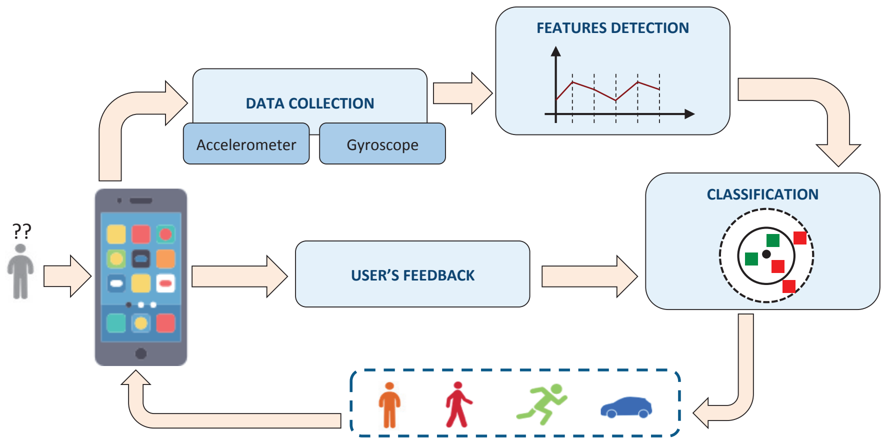


Figure 2.1: System overview for recognizing simple activities. The activities performed by the users are captured by means of the smartphone sensors. Collected data are analyzed to detect some relevant features, that are then used for classification. User's feedback on recognized activities are exploited to improve the classification process.

stricted to specific application scenarios, e.g., e-health, or their inner behavior is unknown.

To better investigate these aspects, this section describes the main components of a sample HAR system able to recognize simple activities, and how these activities can be modeled by means of the motion sensors. Then, the dissertation continues by presenting a more sophisticated algorithm for recognizing complex activities.

2.2.1 A Sample HAR System

In a scenario where the goal is to automatically infer simple human activities, a smart device (e.g users' smartphone) can leverage on the combination of four main components, as depicted in Fig. 2.1. The first is responsible for *data collection*, that is for capturing raw data through the smartphone sensors while an activity is performed. The raw values are sent as input to the *features detection* module, where a set of n-dimensional points are extracted to distinguish different activities. The *classification module* recognizes the activities using a machine learning algorithm and, finally, the user can provide a *feedback* on the output of the recognition in order to allow the system to properly perform future classifications.

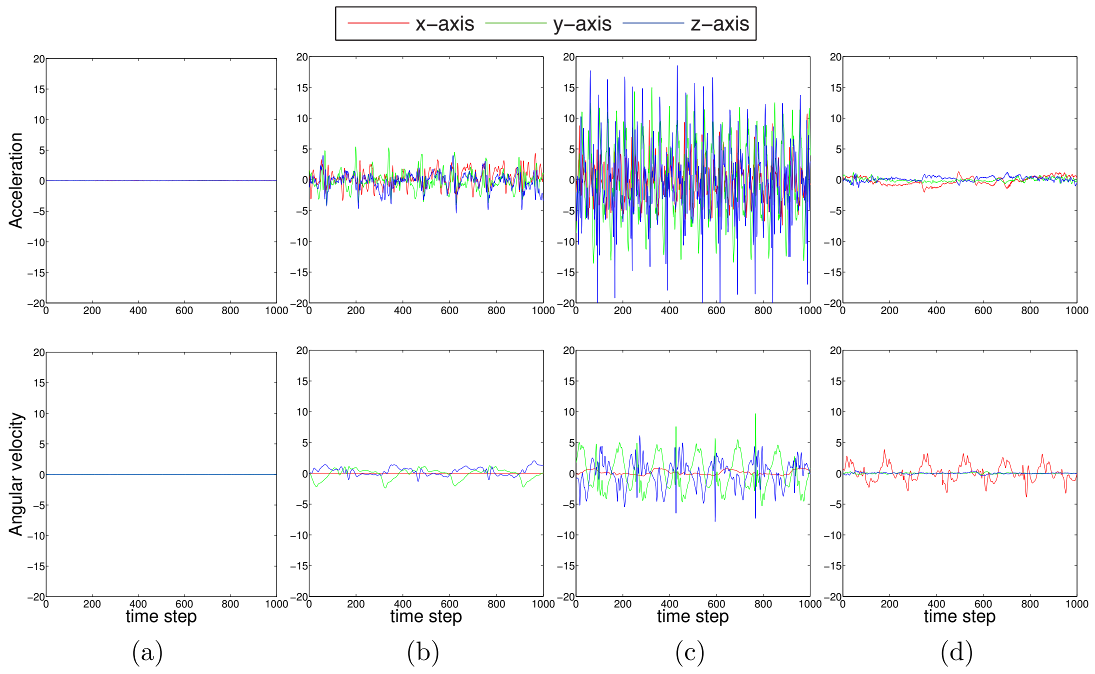


Figure 2.2: Three-axes acceleration (top row) and angular velocity (bottom row) for *still* (a), *walking* (b), *running* (c), and *in-a-vehicle* (d) behaviors.

In order to understand how the activity patterns change according to accelerometer and gyroscope readings, Fig. 2.2 shows values of three-axes acceleration (top row) and angular velocity (bottom row) captured while performing *still*, *walking*, *running*, and *in-a-vehicle* activities, respectively. As regards the acceleration values, even though these four activities look somehow different from each other, some of them, i.e., *still* and *vehicle* share a similar pattern, whilst others, i.e., *walking* and *running* are characterized by high noise as they are intrinsically associated with a significant user movement. On the contrary, by analyzing the values of angular velocity it is possible to notice that *still* and *vehicle* exhibit distinct patterns, whilst other activities are generally characterized by oscillations of different width and frequency. For this reason, the combination of data from the two sensors allows to get the best from both characteristics.

To ensure real-time activity recognition, the input data must be processed within certain time windows in order to extract the features that will be used in

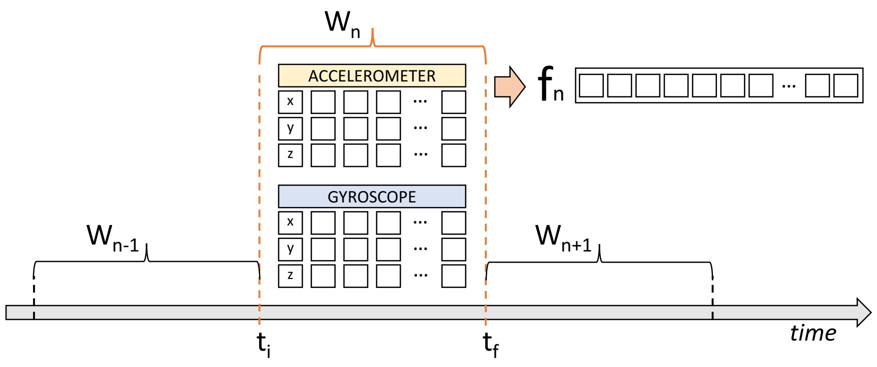


Figure 2.3: Feature extraction mechanism. Accelerometer and gyroscope data are processed within the n-th fixed-length time window W_n, in order to obtain the corresponding feature vector f_n.

the next classification stage. The entire process of feature extraction is shown in Fig.2.3.

An activity ma can be modeled by observing the user behavior from initial time t_i to final time t_f, with $t_i < t_f$. Given that the duration, in seconds, of the activity ma_j is denoted by d_j, data captured within this interval is processed into fixed-length windows of $m \times n$ samples, where m is the number of axes along which measurements are performed. In particular, the activity recognition process is based on (X_A, Y_A, Z_A) values provided by the accelerometer, and (X_G, Y_G, Z_G) values from the gyroscope.

Choosing the proper length of the acquisition window is essential because of the impact it could have on the whole system. Short windows may improve system performances in terms of execution time and CPU load, but may not contain enough information to properly capture the characteristics of the activity. Vice versa, long windows may alter the system performances since information about multiple activities performed in sequence might be analyzed within a single window. Preliminary experiments were performed considering windows of different length without overlap, i.e. from 1 sec up to 5 sec. An overlap between adjacent windows is tolerated for certain applications; however, this is less frequently used in sensor-based HAR applications [Banos et al. (2014)]. While windows of 1 and 2 sec showed low effectiveness in correctly describing an activity, those of 3, 4 and 5 sec were much more significant. For these values, a study was conducted about the time required by smart devices to extract the final feature vector. Results showed that fixed-width windows of 3 seconds are the most proper solution because of the lower processing time [Concone et al. (2017)].

In order to obtain a compact representation of input data, feature vectors are built similar to [Cardone et al. (2013)] by considering (i) *max value*, (ii) *min value*, (iii) *mean*, (iv) *standard deviation*, and (v) *root mean square* over the three accelerometer and gyroscope axes. Therefore, each feature vector f contains 30 elements, i.e., 15 values of acceleration and 15 values of angular velocity.

The classification process is based on the k-nearest neighbors (K-NN) technique. Given the set of feature vectors $(f_1, f_2, \cdots, f_m)$, the key principle behind K-NN is that an unknown feature point f, projected into a *large* training set of labeled data, would be ideally surrounded by samples of its same class. If this happens, the algorithm could assign to f the same class of its closest neighbor, i.e., the closest point in the feature space. More generally, the set S containing the k closest neighbors of f is selected and the most recurrent class in S is assigned to f.

In the simplest scenario, a client/server architecture can be designed, allowing each user to i) share its own data captured by the smartphone, and ii) use the same device to leave feedbacks about the recognition process, indicating, every time an activity has been recognized, whether the output class is correct or not. Incorporating user feedback into the learning process could be a great advantage for a supervised algorithm and, in general terms, for an intelligent system. Feedback could be exploited to select most informative samples [Cohn et al. (2003)], or eliminate noisy ones [Kosmopoulos et al. (2012)] allowing, in turn, to improve classification performance.

In the case of the sample HAR system, data sent by the client are analyzed within the server to determine if the models of the different activities need to be updated. In particular, the feature vector f_{new} received from the client is projected into the current feature space, together with the class declared by the user. Then, f_{new} is compared with existing data from the same class, and if they are similar above a certain threshold, the new feature is temporarily stored in the server, which waits for enough samples to re-compute the activity models and send them back to the client for future classifications.

2.2.2 Recognizing Complex Activities

The sample system described in the previous Section is effective in recognizing simple activities, such as *walking* and *running*. Scaling up from simple to more meaningful activities, e.g. *go-to-work* or *jogging*, such a system becomes less effective because of the adoption of a "basic" machine learning (ML) algorithm that, alone, is not sufficient to fully describe more complex activities.

In order to overcome this limit, a more sophisticated algorithm has been designed and, then, integrated into the proposed framework.

The main idea behind the proposed HAR method is that each feature vector should be able to capture the characteristics of a certain micro-activity ma, i.e. the simple activities described in Section 2.2.1. Thus, a complex activity CA could be seen as a specific sequence of micro-activities $\{ma_1, ma_2, \ldots, ma_n\}$, each performed within one of the n time windows W_n.

Unfortunately, this representation would make it difficult to recognize complex activities of different lengths, and inefficient to recognize long-lasting activities. For this reason, it would be reasonable to find a unique set of Ω relevant micro-activities $\{ma_1, ma_2, \ldots, ma_\Omega\}$, with $\Omega \ll n$, that can properly describe every CA. These Ω elements can be referred as *words* of a *vocabulary*.

This problem was solved by combining k-means [Hartigan and Wong (1979)] clustering and SVM classifiers [Scholkopf and Smola (2001)] to find the set of Ω representative words, and to associate observations with words. These words are then used to train m HMMs, where m is the number of complex activities the system can recognize. This approach, also known as KM-SVM, or CSVM, allows to speed up both the training and the prediction of SVM classifiers on large-scale datasets, and its effectiveness has been discussed in the literature (e.g., [Yao et al. (2013); Vo et al. (2016)]).

Given a set of feature vectors $(f_1, f_2, \ldots, f_n)$, k-means partitions the n observations into Ω sets, $C = (C_1, C_2, \ldots, C_\Omega)$, while minimizing the intra-cluster error. These clusters are used to create a new training set NT, upon which the SVM model will be trained:

$$NT = \{(C_1, T_1), (C_2, T_2), \ldots, (C_\Omega, T_\Omega)\}, \tag{2.1}$$

where the i-th pair (C_i, T_i), with $1 < i < \Omega$, represents the cluster and cluster label, respectively.

SVM is a supervised learning technique that aims to find the best separating hyperplane between two classes according to labeled training samples. Generally, given a training set $X = \{x_1, x_2, \ldots, x_s\}$ and the corresponding label set $Y = \{y_1, y_2, \ldots, y_s\}$, a sample can be expressed as

$$\{x_i, y_i\},\ x_i \in R^d,\ y_i \in \{-1, +1\}, i \in \{1, 2, \ldots, s\}, \tag{2.2}$$

where d is the dimension of the input space, and s is the number of samples. In addition, defining a and b as the weight vector and the bias of optimal hyperplane, respectively, then the separating function can be expressed as

$$ax + b = 0. \tag{2.3}$$

According to this definition, all points belonging to the positive class must satisfy the constraint

$$ax_i + b \geq +1, \quad y_i = +1, \tag{2.4}$$

and the others satisfy

$$ax_i + b \leq -1, \quad y_i = -1. \tag{2.5}$$

Even though SVMs allow to classify samples belonging to two classes, real-world applications usually require to distinguish between a greater number of classes. Multi-class SVMs overcome this limitation by facing the problem through a series of binary SVMs combined according to some strategies (one-versus-all, one-versus-one, and direct acyclic graph); in most cases, the one-versus-one approach is preferable [Duan and Keerthi (2005); Hsu and Lin (2002)]. Assuming that there are exactly Ω classes, one-versus-one multi-class SVMs train a separate classifier for each different pair of classes creating L SVMs, where $L = \Omega(\Omega - 1)/2$. After all classifiers are trained, the classification is done according to *max-win voting* approach.

The output of the process described is a set of words $\{ma_1, ma_2, \dots, ma_\Omega\}$ that combined with each other can be used to model a complex activity. The vocabulary construction is performed on the cloud and is repeated whenever the overall activity models need to be updated. After the new models have been computed-that is, when the vocabulary has been modified, data are sent to all the devices situated in the fog to keep them updated.

The recognition of a new, unknown, activity is performed according to a two-step classification procedure. First, pre-trained SVMs are used to associate each feature vector with the corresponding micro-activity (word) contained in the vocabulary. The second step is based on Hidden Markov Models (HMMs) to model the transitions from one micro-activity to the other.

HMMs [Rabiner and Juang (1986)] are an extension of the Markov chains that aims to find the most probable hidden states according to a sequence of events that can be observed. Unfortunately, in real scenarios, the events are not directly observable and HMMs overcome this limitation introducing hidden events that can be considered as causal factors in the probabilistic model.

Formally, an HMM is totally described by the following quintuple $HMM = (N, M, A, B, \Pi)$, where N is the number of states in the model, M is the number of distinct observation symbols per state, A is the transition probability matrix $\{a_{1,1}, a_{1,2}, \dots, a_{1,N}, \dots, a_{N,N}\}$, B is the emission probabilities matrix $\{b_{1,1}, b_{1,2}, \dots, b_{1,M}, \dots, b_{N,M}\}$, and Π is the initial probability distribution $\{\pi_1, \pi_2, \dots, \pi_N\}$, where the generic π is:

$$\pi_i = P[S_1 = i], 1 \leq i \leq N. \tag{2.6}$$

Finally, being $V = \{v_1, v_2, \dots, v_M\}$ the individual symbols, and q_t the generic state at time t, the transition probability matrix A and observation probability B can be written as:

$$a_{i,j} = P[S_N = j \mid S_{N-1} = i], 1 \leq i, j \leq N, \tag{2.7}$$

$$b_j(k) = P[v_k \; at \; t \mid S_j = q_t], 1 \leq j \leq N, 1 \leq k \leq M. \tag{2.8}$$

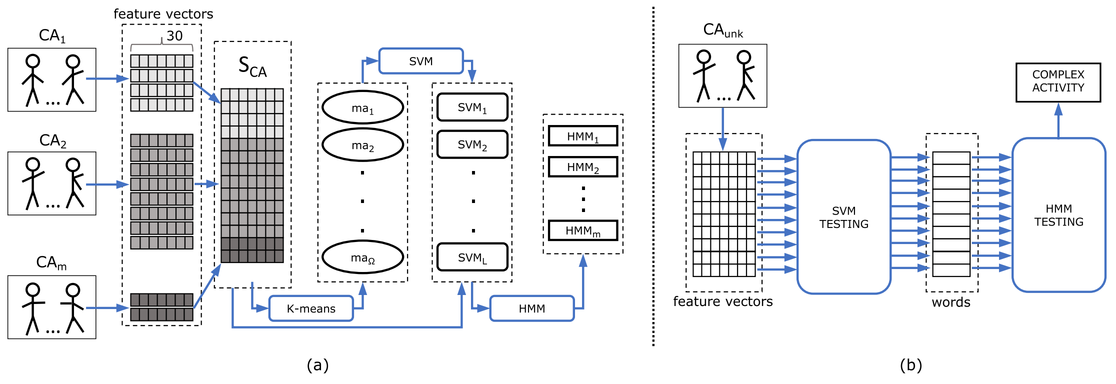


Figure 2.4: The activity recognition process during training (a) and recognition (b) phases.

Generally, when HMMs are used to recognize simple activities, the hidden states are the activities themselves and the observations correspond to sensor data [Kim et al. (2010)]. Given a set of micro-activities y, the problem can be modeled as finding the most likely activity w in a set W:

$$\arg\max_{w \in W} P(w, y). \tag{2.9}$$

By applying the Bayes' Rule, the preceding relation can be rewritten as

$$\arg\max_{w \in W} P(w, y) = \arg\max_{w \in W} \frac{P(y|w)P(w)}{P(y)}. \tag{2.10}$$

Then, the classification of a new, unknown, sequence of micro-activities is performed by testing it against all the HMMs, and selecting the class associated with the largest posterior probability.

Figure 2.4 describes the steps of the proposed HAR algorithm, which can be summarized as follows:

- *Training*

(1) Collect a set S_{CA} containing p repetitions of the m complex activities the HAR system should recognize (note that S_{CA} consists of the feature vectors extracted from raw data).

(2) Apply k-means on S_{CA} to find Ω representative groups of the micro-activities $\{ma_1, ma_2, \ldots, ma_\Omega\}$.

(3) Use data from each group (cluster) to train L SVMs that classify the corresponding micro-activity.
(4) Test each feature vector from the original set S_{CA} against the L SVMs to associate each vector to a word, and represent each CA as a sequence of words.
(5) Use these sequences to train m HMMs.

- *Testing*

(1) Capture a certain unknown complex activity CA_{unk} performed by the user, and represent it as a sequence of feature vectors.
(2) Use the L SVMs to classify each feature vector, translating it to the corresponding word.
(3) Classify the CA, represented as a sequence of words, by means of m HMMs.

To fully understand the steps of the HAR algorithm, consider the complex activity *go to supermarket* (CA_s) that, for a given user, may be composed of an ordered sequence of 5 micro-activities: *going down the stairs* (ma_1); *driving* a vehicle (ma_2); *walking* for a certain amount of time (ma_3); then *driving* a vehicle again (ma_2); *going up the stairs* (ma_4); *staying still* (ma_5) because user is at home.

Before the algorithm is able to recognize CA_s, it must be properly trained, i.e., micro-activities ma_1 to ma_5 must be present in the vocabulary of words. Starting from the raw data extracted from the sensors, the algorithm extracts features and applies k-means clustering and SVMs in succession. The former generates clusters containing similar feature vectors that, in a first step, are used to train SVMs. Then, the second assigns each cluster a unique identifier, thus creating a word of the vocabulary (i.e. the micro-activity). The result will be a sequence of micro-activities that will be used to train the HMMs.

Once the HAR algorithm is trained, let's assume that we want to recognize the activity *go to supermarket*. The first step is to describe in real-time the micro-activities by means of accelerometer and gyroscope sensors embedded in smart devices. To this aim, the sensing device, such as a smartphone, collects raw data from the sensors, process them within fixed-time windows to extracts feature vectors, and test each feature vector against the L SVMs. The role of the SVMs consists in assigning each feature vector the inferred micro-activity, so as to have the ordered sequence that describes the CA_s, e.g. $CA_s = \{ma_1, ma_2, ma_2, ma_3, \ldots, ma_2, ma_4, ma_5\}$. Then, the complex activity is classified by means of HMMs.

2.3 A Distributed HAR Approach Based on Fog-Computing

Fog computing paradigm is particularly suitable for scenarios in which a large number of heterogeneous, ubiquitous, and decentralized devices communicate with each other in order to perform cooperative processing tasks. For this reason, Social Sensing systems can exploit fog computing to design scalable and well performing applications in which humans act as "sensors".

A general crowdsensing process consists of different steps, including low-level data acquisition and analysis, features summarization, intermediate data modeling, high-level reasoning, and so on. Thanks to the latest advances in mobile computing, these tasks do not require to be performed by a single entity, but can be logically distributed among different physical devices cooperating within a common architecture. In particular, the general idea of fog computing is to move early analysis towards the *edge* of the network, while relying on other intermediate (*fog*) or remote (*cloud*) devices for computations of increasing complexity.

Based on the above concepts, this section describes the main features of a human-centric fog-based architecture [Concone et al. (2019)] that can be adopted as basis for a wide number of mobile crowdsensing applications. To this aim, the architecture is composed of three logic levels in which operate devices with increasing computing power.

At the lowest level, see Figure 2.5, edge devices (ED) that are used by many people in their daily lives (e.g., smart wristband devices and smartphones) are responsible for capturing raw data about the user or the environment. In order to reduce bandwidth and power consumption, edge devices do not communicate with each other; still, they are able to locally perform early data preprocessing and send aggregate data to the fog layer.

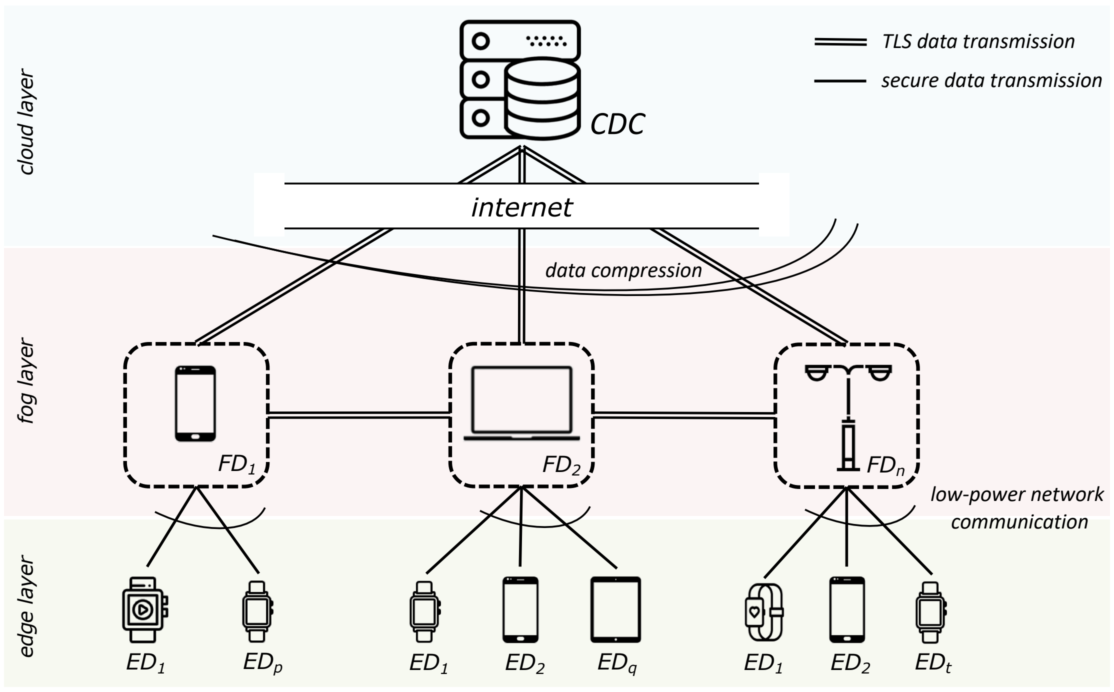


Figure 2.5: Fog-based architecture.

Here, more powerful units, i.e., the fog devices (FD), are suited to perform time consuming tasks, e.g., create mathematical models of environmental measurements taken over time, learn users' habits, recognize relevant pattern, or execute AI algorithms. Each FD can manage a different number of EDs according to its computational power and characteristics. For instance, a smartphone (FD_1 on Figure 2.5) could be used to process data coming from a few wrist-worn devices, while a laptop (FD_2) could be able to manage a greater number of nodes. It would be also possible to consider static fog devices, such as smart poles (FD_n), designed to process data coming from various edge devices owned by moving people.

Edge devices may rely on a larger set of fog entities to accomplish a task: the problem is which one should be selected, and why. This aspect has major implications in heterogeneous wireless networks, such as xG/WiFi, and less in Bluetooth-based communications [Angelakis et al. (2016); Li et al. (2018b)]. In fact, xG/WiFi network are, usually, made up of EDs, FDs, and macro/small-cell base stations that work altogether to achieve the heterogeneous task's deadline in limited available radio and computational resource of the network [Pratap et al.

(2019)]. Hence, the main challenge is how to manage the communication resources, controlled by the base stations, that are needed to transfer information back and forth between edge and fog devices. Thanks to the Bluetooth technology, the proposed framework is beyond this matching problem because an ED is paired with a FD that is the nearby and has not reached the maximum number of available connections, thus bypassing the base-stations. In the worst case, the edge (e.g. smartwatch) is paired with the personal user's smartphone.

Information produced at the edge and fog layers is sent to the Cloud Data Center (CDC) for heavy resource-consuming data analysis and storage. Results of CDC analysis are sent back to the fog in order to make the whole system consistent. The amount of data exchanged between fog and cloud is usually noteworthy, thus compression algorithms can be applied to improve the transmission efficiency.

2.3.1 Application Scenarios

The main purpose of the study conducted here is to present a general fog-based architecture that can be adopted to build a distributed HAR application.

A straightforward solution in mobile scenarios could be to exploit the power capability of users' personal devices, such as smartphones, to process data in the fog. Nevertheless, in a multi-user scenario, a single fog device with a higher level of performance, such as a personal computer, can be used to process and integrate data from multiple devices worn by a community of users. We could also consider different situations in which, for instance, the HAR system exploits information directly captured by users' smartphones. Here, these devices would be logically located at the bottom layer of the architecture, while the fog layer could consist of other types of units.

The generality of the proposed architecture allows to use at the fog layer any device with enough computing power to perform raw data analysis and send aggregated data to the cloud. For instance, a HAR system based on video sensors could be implemented by means of RGB/RGBD cameras (sensing layer) sending raw data to some local processing units responsible for performing activity recognition (fog layer), and then to remote storage and synchronization centers (cloud layer). Moreover, to carry out the HAR process in more complex scenarios, devices at the fog layer can share information with each other. For example, three situations where fog-to-fog communication can be effective are the following:

- *Alerting*: Monitoring the user's activities in critical environments would allow timely detection of dangerous situations. In such a scenario, the output of the HAR process performed by a fog device could be used to send prompt alerts to other devices at the same layer, thus bypassing the cloud. For instance, in a nursing home or in a factory, the activities can be recognized by means of wrist-worn devices (one per user), while some PCs (e.g., one per environment) can be used at the fog layer. The detection of unexpected behaviors could be immediately notified to other fog devices, without any cloud intervention, enabling a prompt response of the security staff. From an architectural point of view, this can be easily implemented by providing the fog devices responsible for HAR with an additional software module specifically designed to handle the alerting procedures.

- *Distributed and continuous tracking*: The proposed HAR technique aims at recognizing complex activities of different duration that can be performed in different places. Some of the considered activities are made of dynamic (e.g., walking, running) and static (working at a PC) phases. In such a composite scenario, we can imagine a fog layer made of wearable mobile devices (e.g., smartphones) to perform activity recognition during the dynamic phase and stationary devices (e.g., PCs) to continue the recognition once the user reaches a static place (e.g., the office). To this aim, fog devices must be able to share with each other information about the micro-activities performed at a given time, so as to build the overall sequence that describes the complex activity. From an architectural point of view, this can be obtained by providing fog devices with the capability of discovering themselves and pairing to each other automatically.

- *Health promotion*: In a collaborative scenario, devices at the fog layer can interact to motivate the users to achieve a certain result. For instance, if we consider a community of people doing sports in the same place (a gym, a rehabilitation center, etc.), several fog devices could recognize the activities performed by one or more users and share related information (e.g, elapsed time, speed, calorie consumption, heart rate) with the community to stimulate the users in achieving their goals.

In addition to the situations described so far, the generality of the proposed fog architecture enables even more generic and complex scenarios, in which information about user activities may be combined with other relevant contextual information. For instance, in a smart campus, where there are thousands of students who perform various daily activities (e.g. attending classes, having lunch, using sports facilities, studying in the library), the combination of user activities, timestamp and GPS position may allow an intelligent system to analyze the flow of people currently on campus, and to identify trends and behavioral patterns on many factors of primary interest. Also in this case, the fog devices are employed for performing the HAR process, while the CDC may be used for more complex tasks, such as retrieving and managing the users' behavioral patterns. Then, the CDC could modify and improve the quality of services offered to students. For example, the intelligent system may suggest that administrators change the times or frequency of the campus shuttle buses to suit the expected influx of students during a certain peak time. All this could be done in real-time and on the basis of collected data grouped by day of the week, period of year and time of day. In addition, if on certain days of the week some sports facilities are typically left unused after a certain time, the opening and closing times can be optimized.

2.3.2 Case Study: Recognizing Complex Activities Performed by a Community of Users

To validate the effectiveness of the proposed solution, a straightforward case study in which users may wear smart devices while performing some activities is investigated. A reasonable assumption in such a scenario could be to exploit smartwatches to track the users' movements, passing heavy data processing to more powerful fog devices, such as the users' smartphones (Figure 2.6).

The recognition scheme based on k-means, SVMs, and HMMs allows to easily extend the recognition capability of the system (e.g., by including a larger number activities), without the need for redesigning the other components of the system.

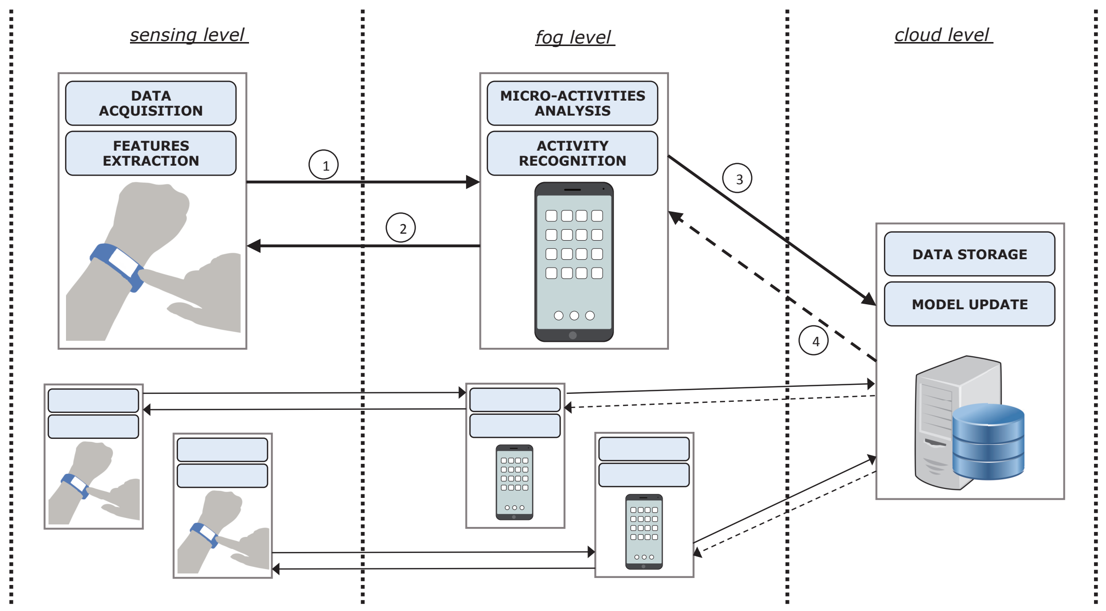
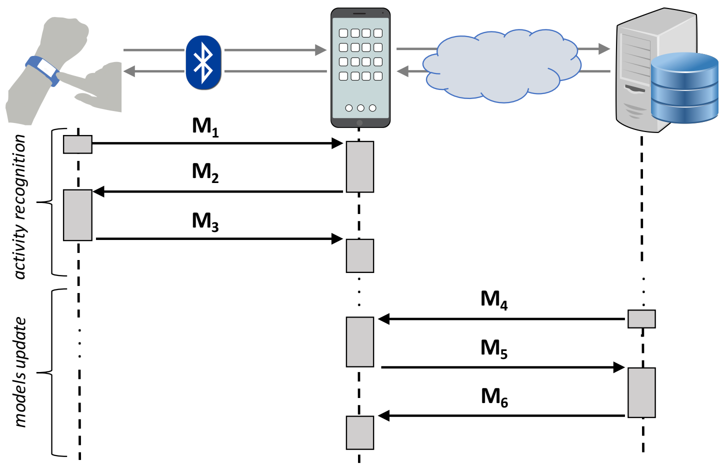


Figure 2.6: HAR through users' personal devices. Wrist-worn devices are responsible for capturing sensory data and summarizing relevant information. The extracted features are transmitted to the user's smartphone (1), where data are analyzed to detect sequences of relevant micro-activities and recognize the complex activity performed by the user (2). Data are temporarily stored in the smartphone and then sent to the cloud (3), where the system parameters and models are updated and sent back to the devices (4).

With regards to the type of activities to recognize, it is considered a set of complex activities that can be reasonably decomposed in simple, atomic micro-activities. For instance, for a given user, the everyday activity *go to work* may consist of a sequence of *walking* for a while, driving or being in a *vehicle* for a certain amount of time, then *walking* again, going up the *stairs*, and finally arriving at the office staying *still*. All of these phases can be traced by the sensors on edge of the network, such as those embedded in the smartwatches. To recognize a set of known micro-activities, each sensing device collects data from accelerometer and gyroscope sensors, extracts a feature vector for each time window, and sends the set of feature vectors to the fog devices.

The fog devices of the considered case study (i.e., the users' smartphones) recognize the performed micro-activity (e.g., *walking*) producing the corresponding *word* of the vocabulary. Fog devices also act as buffers for temporarily storing



M_1:	*the edge sends feature vectors to the fog*
M_2:	*the fog sends the recognized activity to the edge device*
M_3:	*the edge sends HAR data, e.g. feature sets and user feedback*
M_4:	*the cloud checks for new available data*
M_5:	*the fog sends the buffered data*
M_6:	*the cloud sends the new models to the fog*

Figure 2.7: Information exchanged between the smart devices and the cloud.

feature vectors, users' feedback, and any other data that need to be transferred to the cloud. The CDC consists of a server that analyses the different models coming from the smartphones to maintain a unique set of known activities and refines the local models at the fog layer.

The entire dataflow through the layers of the proposed architecture, from the sensing devices to the smartphones to the cloud and back, is summarized in Figure 2.7.

During the first phase (i.e., the activity recognition), the wrist-worn device creates a message M_1 containing the extracted feature vector and sends it to the smartphone through a Bluetooth connection. Message M_1 is received and parsed by the smartphone that associates a particular word of the dictionary to each feature vector by means of SVMs. This process continues until the smartphone has enough words to build a sequence. Once the sequence of words is completed, the HMM classification is performed and the recognized activity is provided as output. This information, contained in the message M_2, is received by the user on its wrist-worn device. According to the quality of the recognition, the user can give a positive or negative feedback through the message M_3. Data collected so far, such as feature sets and user feedback, are stored in the smartphone, ready to be sent to the cloud when requested.

During the second phase, the cloud server sends a message M_4 to check if new data are available. If yes, the smartphone sends the message M_5 containing all data buffered during the previous phase, deleting them from local memory. At this point, the server processes the incoming information so as to evaluate if it is necessary to update the dictionary and the HMM parameters. Once that the models have been updated, a message M_6 is sent to the smartphones to synchronize them.

Chapter 3

Security Issues

In Social Sensing applications, given the central role of people in the sensing process, preserving sensitive information is mandatory. Especially during the data collection and sharing phases, personal devices can potentially reveal private information, seriously compromising the privacy of end users. For example, an application in an e-health scenario could share the heart rate, stress level or oxygen level in the blood revealing particularly sensitive information to a malicious entity or an interested third party.

These issues become even more complex when considering the most recent distributed computing paradigms, such as the one adopted in this thesis. In fact, the fog computing was born as an extension of cloud computing inheriting all its security and privacy issues and adding new ones because of its distributed nature. Heterogeneity in fog node and fog network, requirement of mobility support, massive scale geo-distributed nodes, location-awareness and low latency represent only a few examples of the challenges introduced by fog-based infrastructures [Yi et al. (2015)].

As regards the distributed crowdsensing scenario discussed here, the main requirements in security consist in (i) allowing edge devices to perform their tasks without revealing the user's identity, and (ii) securing the data exchanged between entities in the fog architecture to avoid any attackers to steal private information. This must also take into account the constrained capabilities of the edge devices, which are limited in computation and resources. Hence, a comprehensive security mechanism needs to be designed to protect both data stored on the physical devices and messages exchanged within the fog architecture, paying attention to the overhead introduced in making the system secure.

This part of thesis focuses on the design of a secure mobile crowdsensing protocol that exploits two lightweight encryption techniques, i.e. Elliptic-Curve Cryptography and Extended Triple Diffie-Hellman Key Agreement, that are particularly suited for low-power mobile devices. The protocol allows to secure five phases that are common to many crowdsensing systems in which a fog-based approach is adopted; namely, the *fog* and *edge enrollment* phases allow edge and fog devices to mutually prove their identity and to certify their legitimacy within the system; the *key agreement* makes the two parties able to build a secure communication channel; during the *edge-fog communication* phase the messages exchanged to support the crowdsensing algorithm are protected from external attacks; finally, the *fog-cloud communication* allows to maintain the overall models created by the system.

3.1 Existing approaches

Sensor-based crowdsensing has been widely addressed in the literature because of its suitability for different application scenarios. Simple mobile participatory sensing applications can be developed by relying on existing software libraries, such as the Google Activity Recognition APIs and the IOS Core Motion framework. However, when the goal is to perform more intensive tasks, such as real-time human activity recognition, many works highlighted the need for frameworks in which multiple devices, e.g., mobile devices with limited resources, are able to cooperate with each other. In this context, fog computing paradigm [Bonomi et al. (2012)] can be adopted to realize well-performing systems that are able to process large amounts of data *locally*, and *timely*. The general idea behind a fog-based monitoring system is to distribute *data collection*, *analysis*, and *storage* tasks among different devices located at distinct logic levels. Due to the limited computational capabilities and heterogeneity of fog devices, as well as the dynamic nature and unpredictability of fog environments, task allocation is a challenging problem that has been widely addressed in the literature [Ghobaei-Arani et al.

(2019)]. In [Ghobaei-Arani et al. (2020)], for instance, task scheduling is performed by means of an efficient moth-flame optimization algorithm that guarantees quality of service in Fog-based cyber-physical systems.

Many works have adopted fog-based approaches to support distributed sensing and processing. In [Jayaraman et al. (2014)], for instance, is a platform aimed at supporting mobile crowd-sensing applications in a smart city. The system proposed in [De Paola et al. (2020)] relies on a fog-based architecture to realize the different components of an Ambient Intelligence (AmI) system aimed to achieve energy efficiency in smart buildings. In [Perera et al. (2015)], Internet Connected Objects (ICOs) are used at the edge of the network to measure air quality, users' movements, and sounds, while user smartphones are exploited as intermediate gateways. In [Cao et al. (2015)], a fog-based platform designed to detect users' falls in a e-health scenario is described, while a comprehensive review of fog-based IoT systems and technologies for healthcare is presented in [Mutlag et al. (2019)]. Beside being the core of the application-oriented solutions discussed so far, machine learning techniques are also commonly used to deal with technical aspects of fog computing, such as efficient resource management, latency and energy consumption reduction [Oma et al. (2018)], as well as modeling network traffic. An exhaustive analysis of machine learning in the context of fog computing is proposed in [Abdulkareem et al. (2019)].

The high degree of modularity of fog systems makes them also prone to cyber security attacks that can be performed against every element of the infrastructure [Roman et al. (2018); Khan et al. (2017)]. Nevertheless, most of the works presenting novel fog-based applications are focused on the description of the edge-fog-cloud infrastructure and the corresponding service deployment; as a consequence, in such papers security issues are frequently omitted or ignored. Other works presented ad-hoc security schemes that cover some specific threats only; then, many challenges still remain open [Zhang et al. (2018)]. For instance, two important security issues concern *authentication* and *trust* between the different entities that cooperate within a fog network.

Whereas *authentication* can be efficiently achieved through soft and behavioral biometrics [Alqarni et al. (2020)], as well as by implementing lightweight schemes [Wazid et al. (2019)] that meet the low computational capabilities of edge/fog devices, ensuring *trust* in a dynamic fog environment is usually harder because the behavior of mobile nodes is likely to change over time. In [Amor et al. (2017)], authors describe a Secure Fog-based Communication Scheme (SFCS) to ensure trust between edge devices and fog devices via a novel session key agreement. Such a protocol uses ECC encryption, Discrete Logarithm Problem and bilinear pairing allowing to establish the session key without extra-parameters. Octopus [Ibrahim (2016)] is an authentication scheme for fog-based architectures that allows any edge device and fog device to mutually authenticate each other by means of a long-lived secret key, and symmetric encryption. The authors of [Li et al. (2018a)] propose a forwarding scheme for mobile IoT devices in which the trustworthiness of the nodes is estimated according to their service degree, and then used to build reliable communities. Other works address only secure edge-fog communications, without considering the security issues of fog-cloud data transmission.

Authentication and *trust* are strictly connected to safeguarding of data; conversely, one of the most important requirements for any application that exploits people's participation is *privacy*, i.e, safeguarding of user information. In [Xu et al. (2019)], for instance, the authors propose a novel double-masking protocol that guarantees the authenticity of data shared by multiple users while protecting their privacy. Distributed consensus in the context of the Internet of Vehicles is achieved in [Bonadio et al. (2020)] through *permissionless* blockchains in order to guarantee reliability of information collected from individual vehicles. Similarly, blockchain technology is adopted in [Hu et al. (2020)] both to ensure privacy during the mobile crowdsensing process, and to protect the reward mechanism for participants.

More generally, privacy can be achieved in a number of ways; for instance, it would be crucial to avoid the fog entities to access data coming from individual edge devices. To this aim, *attribute-based encryption* [Zhang et al. (2017)], *data perturbation* [Chamikara et al. (2018)] and *privacy-preserving data aggregation* [Lu et al. (2017)] techniques can be adopted.

For instance, the privacy leakage problem associated with accelerometer data sharing are highlighted in [Xiao et al. (2018)] through an information-aware visualization tool. The authors of [Hu et al. (2017)] focus on security and privacy of a face identification and resolution framework (FIRF) that exploits fog entities to reduce the processing time and reduce the bandwidth usage. To this aim, authentication and key agreement schemes are presented to protect the framework from some well-known attacks, e.g., man-in-the-middle, eavesdropping, and data hijack. FIRF provides adequate level of security in fog-cloud communications, while ignoring the protection of data shared between edge and fog devices. A more specific Anonymous and Secure Aggregation Scheme (ASAS) for fog-based applications is presented in [Wang et al. (2018)]. The main goals of ASAS are protecting the identities of edge devices by using pseudonyms, and guaranteeing data secrecy by means of robust homomorphic encryption (HE). Unfortunately, this last feature makes ASAS unsuitable for real-time applications because of the HE computational complexity [Moore et al. (2014)]. In [Wazid et al. (2019)] a user authentication and key management scheme for fog computing services, called *SAKA*, is presented. SAKA exploits a combination of lightweight cryptographic techniques (i.e., one-way hash functions, bitwise exclusive-OR, and elliptic curves) so as to meet the computational requirements of resource constrained devices. Authors show that, compared to other protocols, their solution provides better performances and security features with a low overhead. The security and privacy challenges discussed so far are illustrative, but not exhaustive, of a class of problems that need to be faced during the design of a fog-based system. A comprehensive literature review on the security challenges in fog-computing is provided in [Yakubu et al. (2019)].

3.2 Requirements and Tools for Lightweight Security

In general terms, private data must be protected from two classes of attackers: internal and external. The former are authorized users that aim to infer information about other users, or the inner system behavior. The latter operate from outside the perimeter of the system and are in most cases easier to detect. *Impersonation*, for instance, is a type of attack in which the intruder I assumes the identity of a legitimate user L. In a mobile crowdsensing scenario, for instance, an impersonation attack could allow I to obtain a valid session key to transmit forged information, e.g., corrupted data or wrong feedbacks, that would compromise the performance of the system. Similarly, *Man-in-the-Middle* (MitM) attack, i.e., a simultaneous double impersonation attack, could be performed to alter the communication between two parties, while making both believe that are directly communicating with each other. In the context of IoT and smart devices, another class of attacks, named *replay* attacks, can be performed to retransmit the same data over and over in order to cause a Denial of Service (DoS) on the target, e.g., the fog device.

In order to contrast these threats, a secure mobile crowdsensing system should meet the following security requirements:

- **registration to the service**: all the entities must be registered with the system before they can use the services;
- **data secrecy and integrity**: neither internal nor external attackers must be able to read and modify the content of data stored within the devices, and the messages exchanged between the entities;
- **timeliness**: an attacker can not intercept messages within a valid session and retransmit them at a later time;
- **user privacy**: the fog nodes must be able to process information sent by the edge, e.g., in order to perform activity recognition. However, this process must be performed respecting the privacy of the user, e.g., the fog must infer the activity performed without being able to associate it to a particular user.

Moreover, given the low computation capability of mobile devices, the overhead introduced by the cyber security mechanisms and protocols should be quite limited. When relying on a symmetric encryption algorithm, for instance, an initial overhead is introduced because all parties involved in data transmission have to agree on a secret key before data can be actually sent through a secure communication channel.

Key exchange protocols based on RSA (Rivest-Shamir-Adleman) and Diffie-Hellman (DH) have been extensively adopted in the literature. However, these techniques require a considerable computational effort that makes them unsuitable for devices with limited resources. For this reason, more recently, light encryption techniques that are more appropriate for IoT and mobile devices have been proposed.

3.3 Efficient Key Exchange on Mobile Smart Devices

One of the most convenient approach for efficient asymmetric cryptography is the Elliptic-Curve Cryptography (ECC). The main benefit provided by ECC is the use of smaller keys than other algorithms, while achieving the same level of security [Alvarez et al. (2017)]. This feature is crucial to reduce the encryption time, especially when dealing with huge amount of data. In [Al Hamid et al. (2017)], for instance, big data collected in healthcare systems are efficiently secured by means of bilinear pairing cryptography. In the scenario addressed here, ECC allows to significantly reduce the time required for generating and sharing the keys, while also minimizing the storing space and the power consumptions.

Different types of elliptic curves [Bos et al. (2014)], and many standards (such as NIST FIPS 186-2 [Nist (1999)] and IEEE P1363 [Group et al. (2005)]) have been presented in the literature. However, both academia and industry have recently started to adopt non-standard curves that guarantee a greater level of security, while also reducing the computation time. *Curve25519*, for instance, has been proven to be resistant to timing attacks and twice as fast as standard curves [Bernstein (2006)]. All parties that want to use ECC must agree on a set of values that defines the elliptic curve, i.e., the *domain parameters*. For a generic curve, this set is indicated as $D = \{q, FR, a, b, G, n, h\}$, where q is an integer specifying the finite field $\mathbb{F}_q$, FR indicates the basis used for representing the field elements, $\{a, b\} \in \mathbb{F}_q$ define the equation of the elliptic curve, G is a distinguished point of order n in an elliptic curve group, and h represents the cofactor.

Elliptic-Curve Cryptography is the basis for the Extended Triple Diffie-Hellman Key Agreement (X3DHKA) that is currently adopted as security mechanism in several Android and IOS applications. Considering two parties A and B, the X3DHKA protocol consists of three phases and makes use of the following keys:

- $\mathsf{PU_A}$, $\mathsf{PU_B}$: long-term public keys of the two entities;
- $\mathsf{EPU_A}$, $\mathsf{EPR_A}$: an ephemeral key pair used by A;
- $\mathsf{SPK_B}$: a momentary prekey signed by B that will be updated at some interval (e.g. once a week, or once a month);
- $\mathsf{OPK_{Bi}}$: a set of one-time prekeys used by B.

X3DHKA is designed for both asynchronous and synchronous settings. In the first scenario, B sends to a Key Management Server (KMS) the long-term public key $\mathsf{PU_B}$, the signed prekey $\mathsf{SPK_B}$, the digital signature on $\mathsf{SPK_B}$, and a set of m one-time prekeys $S = \{\mathsf{OPK_{B1}}, \mathsf{OPK_{B2}}, \ldots, \mathsf{OPK_{Bm}}\}$.

Then, A asks KMS for the bundle provided by B. The B's prekey signature is checked and the protocol is stopped if the verification fails; otherwise, A generates an ephemeral key pair $(\mathsf{EPU_A}, \mathsf{EPR_A})$. The 32 bytes session key $\mathsf{K_S}$ is obtained by calculating the following functions:

$$\mathsf{KDF}(\mathsf{DH}(\mathsf{PU_A}, \mathsf{SPK_B}) || \mathsf{DH}(\mathsf{EPU_A}, \mathsf{PU_B}) \; || \; \mathsf{DH}(\mathsf{EPU_A}, \mathsf{SPK_B}) \; || \; \mathsf{DH}(\mathsf{EPU_A}, \mathsf{OPK_B})),$$

where KDF is the HMAC-based key derivation function described in [Layer (2017)], and the generic $\mathsf{DH}(\mathsf{PU}_1, \mathsf{PU}_2)$ is an Elliptic Curve Diffie-Hellman function, e.g., *Curve25519*, that involves the public keys PU_1 and PU_2.

Finally, A sends a message to B containing the keys $\mathsf{PU_A}$, $\mathsf{EPU_A}$, the value i that identifies the one-time prekey $\mathsf{OPK_{Bi}}$ used by A, and an initial message encrypted with the session key $\mathsf{K_S}$. The entity B will use such information to calculate the DH and KDF functions and derive the key $\mathsf{K_S}$. Once the session key has been generated, the one-time keys are deleted to guarantee forward secrecy.

Conversely, X3DHKA can be executed in a synchronous way by letting B directly communicate with A to request necessary information and establish secure communication.

3.4 Making Mobile Crowdsensing Secure

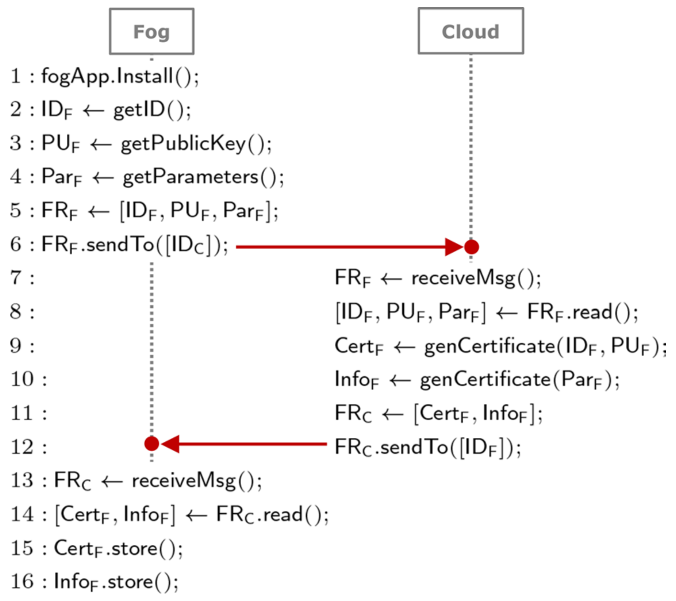
In order to provide the end user with the functionalities discussed so far, the devices deployed at the edge, fog, and cloud layers must be provided with three different software components.

The core of the platform resides on the Cloud Data Center; here, besides performing the specific crowdsensing algorithms, the software is responsible for managing the devices that operate at the underlying layers. In particular, one of the most important roles of the CDC is to supervise the registration of new edge and fog devices by providing them with their corresponding apps.

The *fogApp* installed on FDs makes them able to perform time-consuming tasks (e.g., activity recognition), to announce the presence of fog devices to the edge devices, and to establish secure connections with other devices at the fog layer. Any new FD demanding to be part of the system is required to install the *fogApp* first.

In a similar way, the end users willing to participate to the system need to install on the edge device the *edgeApp*. At first, a light version of this app supporting only the discovery of nearby fog devices is installed. Other functionalities (HAR algorithms and cryptographic suite) will be available just after the *edge registration* has been completed. This phase starts with a fog announcement/discovery step that allows to create a preliminary, secure, association between the edge and fog layers. By means of this channel, the edge exploits the fog as intermediary with the cloud in order to obtain the full version of *edgeApp*, and complete the registration procedure.

Anytime an edge device approximates to a new fog device, the same announcement/discovery procedure is followed to create a new edge-fog association. For instance, an edge device ED_1 could have obtained the *edgeApp* by registering with the fog devices FD_1, but a new association can be created later for transferring data to a different device FD_2. In such a scenario, ED_1 and FD_2 have to agree on a session key aimed at protecting their communications. Moreover, the cloud may act as a Key Management Server providing all the FDs with a session key for fog-to-fog communications. This key is initially included in the *fogApp*, and regularly updated. In particular, in order to meet the computational capabilities



```
1 : fogApp.Install();
2 : $ID_F$ ← getID();
3 : $PU_F$ ← getPublicKey();
4 : $Par_F$ ← getParameters();
5 : $FR_F$ ← [$ID_F$, $PU_F$, $Par_F$];
6 : $FR_F$.sendTo([$ID_C$]);
7 :                         $FR_F$ ← receiveMsg();
8 :                         [$ID_F$, $PU_F$, $Par_F$] ← $FR_F$.read();
9 :                         $Cert_F$ ← genCertificate($ID_F$, $PU_F$);
10 :                        $Info_F$ ← genCertificate($Par_F$);
11 :                        $FR_C$ ← [$Cert_F$, $Info_F$];
12 :                        $FR_C$.sendTo([$ID_F$]);
13 : $FR_C$ ← receiveMsg();
14 : [$Cert_F$, $Info_F$] ← $FR_C$.read();
15 : $Cert_F$.store();
16 : $Info_F$.store();
```

Figure 3.1: Fog devices enrollment.

of the smart devices considered here, two lightweight encryption techniques, i.e., Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES), are used.

The devices discussed so far need to continuously communicate with each other in order to accomplish their tasks. In general terms, all network activities that involve two or more communicating remote entities are governed by a protocol. A protocol defines the format and the order of messages exchanged between two or more parties, as well as the actions taken on the transmission and/or receipt of a message or other event [Layer (2017)]. The next sections describe the entities, the messages exchanged, and the actions needed to implement the SMCP protocol.

3.4.1 Fog enrollment

A new fog device can be added to the system by installing on it the *fogApp* (Fig 3.1 - line 1). In order to support secure communications, this app includes the elliptic curve domain parameters D, and the cloud long-term public key PU_C; starting from them, the fog device generates a long-term key pair $\{PU_F, PR_F\}$.

The fog registration procedure starts with preparing and sending the message $\mathsf{FR_F}$ (line 6) that includes the identifier of the fog ($\mathsf{ID_F}$), the set of parameters describing the characteristics of the device ($\mathsf{Par_F}$), and the fog public key ($\mathsf{PU_F}$). During the registration, two certificates are generated by the cloud (lines 9-10). The former, $\mathsf{Cert_F}$, contains the values $(\mathsf{ID_F}, \mathsf{PU_F})$ and will be used by any edge device to recognize that fog as legitimate. The latter, $\mathsf{Info_F}$, ensures the edge for the authenticity of the computational parameters contained in $\mathsf{Par_F}$. Without the $\mathsf{Info_F}$ certificate, an attacker could exhibit a forged $\mathsf{Par_F}$ to monopolize the communications with the neighboring edges. The registration confirmation is sent to the fog device with the message $\mathsf{FR_C}$ (line 12).

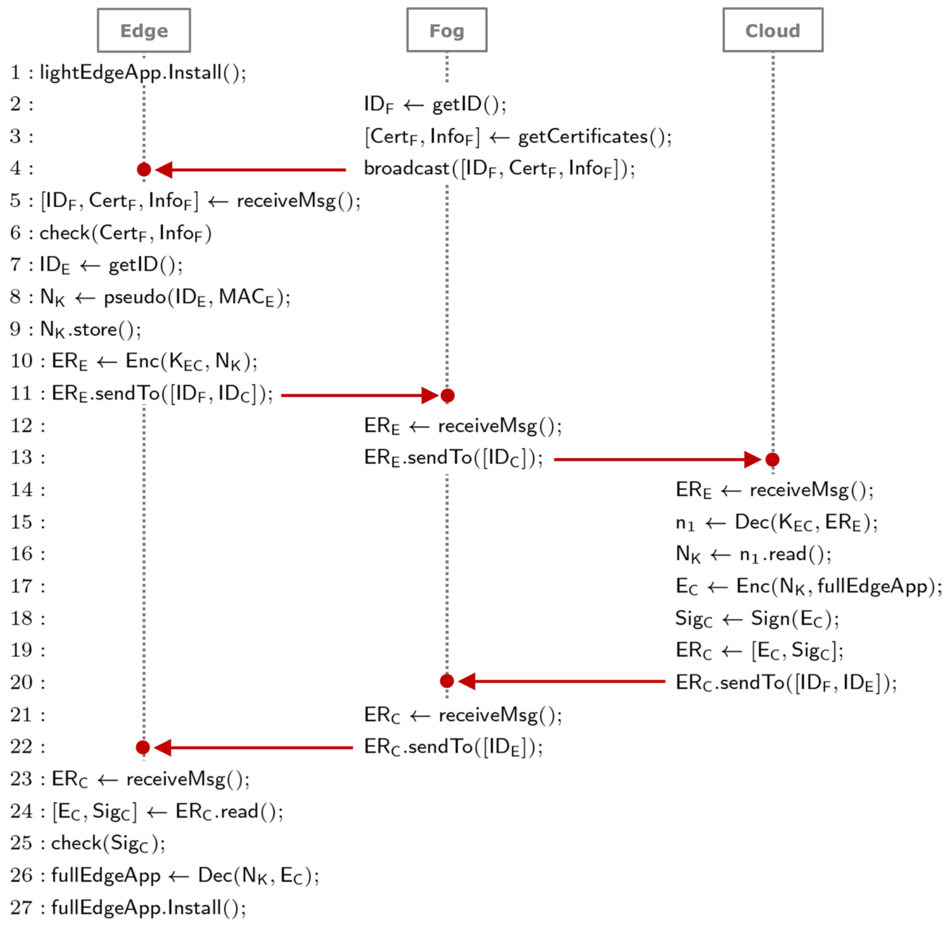
The computational capabilities of fog and cloud devices make them able to implement the TLS protocol; thus, both $\mathsf{FR_F}$ and $\mathsf{FR_C}$ can be transmitted securely over a TLS channel preventing external attacks, such as MitM, Replay, or Impersonation. Once the registration is completed, each fog device will own a long-term key pair $\{\mathsf{PU_F}, \mathsf{PR_F}\}$, and the pair of certificates $\{\mathsf{Cert_F}, \mathsf{Info_F}\}$ (lines 15-16).

After registering to the system, the fog device is able to generate all the parameters needed to communicate with the edge units, according to the X3DHKA scheme described in Section 3.3.

3.4.2 Edge enrollment

In order to take advantage of the services offered by the system, the end user also needs to install a mobile application on its edge device. The light version of the *edgeApp* (Figure3.2 - line 1) includes the fog discovery routines, a symmetric key $\mathsf{K_{EC}}$ for early Edge-Cloud communications, and the cloud public key $\mathsf{PU_C}$.

In an early phase, the fog announces its presence by broadcasting a message containing the fog $\mathsf{ID_F}$ and the pair of certificates obtained during the fog registration step (line 4). Such information is used by the edge to verify the identity of the fog (line 6), and to send the request for downloading the full version of the *edgeApp* (line 11). To this aim, the edge sends to the CDC (by relying on the fog $\mathsf{ID_F}$) the message $\mathsf{ER_E}$ (line 13) containing a pseudorandom number computed as function of the edge identifier $\mathsf{ID_E}$ and the physical address of the device.



```
1 : lightEdgeApp.Install();
2 :                                   $ID_F \leftarrow$ getID();
3 :                                   $[Cert_F, Info_F] \leftarrow$ getCertificates();
4 :                                   broadcast($[ID_F, Cert_F, Info_F]$);
5 : $[ID_F, Cert_F, Info_F] \leftarrow$ receiveMsg();
6 : check($Cert_F, Info_F$)
7 : $ID_E \leftarrow$ getID();
8 : $N_K \leftarrow$ pseudo($ID_E, MAC_E$);
9 : $N_K$.store();
10 : $ER_E \leftarrow$ Enc($K_{EC}, N_K$);
11 : $ER_E$.sendTo($[ID_F, ID_C]$);
12 :                                  $ER_E \leftarrow$ receiveMsg();
13 :                                  $ER_E$.sendTo($[ID_C]$);
14 :                                                      $ER_E \leftarrow$ receiveMsg();
15 :                                                      $n_1 \leftarrow$ Dec($K_{EC}, ER_E$);
16 :                                                      $N_K \leftarrow n_1$.read();
17 :                                                      $E_C \leftarrow$ Enc($N_K$, fullEdgeApp);
18 :                                                      $Sig_C \leftarrow$ Sign($E_C$);
19 :                                                      $ER_C \leftarrow [E_C, Sig_C]$;
20 :                                                      $ER_C$.sendTo($[ID_F, ID_E]$);
21 :                                  $ER_C \leftarrow$ receiveMsg();
22 :                                  $ER_C$.sendTo($[ID_E]$);
23 : $ER_C \leftarrow$ receiveMsg();
24 : $[E_C, Sig_C] \leftarrow ER_C$.read();
25 : check($Sig_C$);
26 : fullEdgeApp $\leftarrow$ Dec($N_K, E_C$);
27 : fullEdgeApp.Install();
```

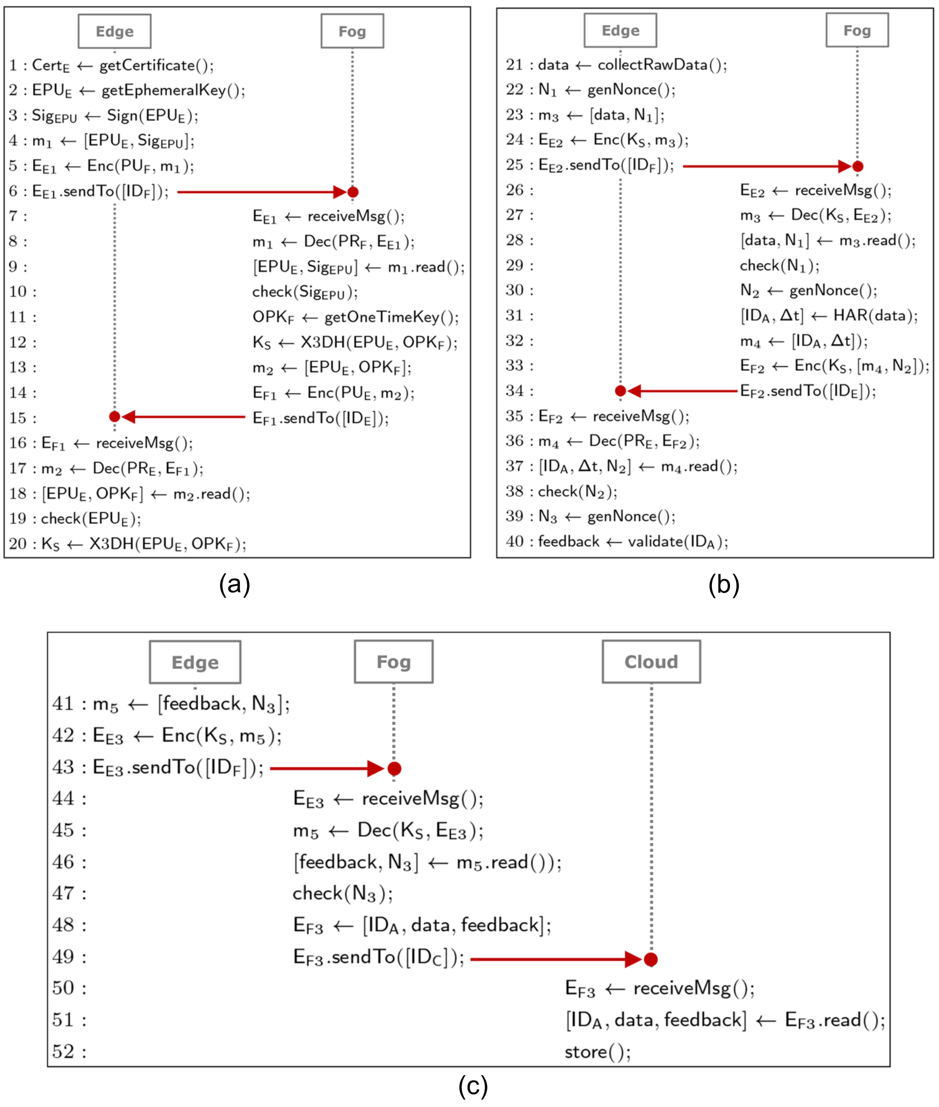
Figure 3.2: Edge devices enrollment.

After receiving $\mathsf{ER_E}$, the CDC sends the logic set of messages $\mathsf{ER_C}$ containing the fullEdgeApp, which is signed, encrypted, and sent to the edge $\mathsf{ID_E}$ through the fog $\mathsf{ID_F}$ (lines 18-20).

In addition to the fog and edge enrollment, SMCP covers all the phases in which the crowdsensing algorithms are executed actually.

3.4.3 Protect exchanged data during HAR phases

To this aim, the edge device sends the message $\mathsf{E_{E1}}$ to the fog containing the key $\mathsf{EPU_E}$ and the digital signature on $\mathsf{EPU_E}$ (Figure 3.3(a), line 6). Ciphering $\mathsf{EPU_E}$ with the public key of the fog prevents an intruder to know the seed from which it



```
1 : $Cert_E \leftarrow getCertificate();$
2 : $EPU_E \leftarrow getEphemeralKey();$
3 : $Sig_{EPU} \leftarrow Sign(EPU_E);$
4 : $m_1 \leftarrow [EPU_E, Sig_{EPU}];$
5 : $E_{E1} \leftarrow Enc(PU_F, m_1);$
6 : $E_{E1}.sendTo([ID_F]);$
7 :                    $E_{E1} \leftarrow receiveMsg();$
8 :                    $m_1 \leftarrow Dec(PR_F, E_{E1});$
9 :                    $[EPU_E, Sig_{EPU}] \leftarrow m_1.read();$
10 :                   $check(Sig_{EPU});$
11 :                   $OPK_F \leftarrow getOneTimeKey();$
12 :                   $K_S \leftarrow X3DH(EPU_E, OPK_F);$
13 :                   $m_2 \leftarrow [EPU_E, OPK_F];$
14 :                   $E_{F1} \leftarrow Enc(PU_E, m_2);$
15 :                   $E_{F1}.sendTo([ID_E]);$
16 : $E_{F1} \leftarrow receiveMsg();$
17 : $m_2 \leftarrow Dec(PR_E, E_{F1});$
18 : $[EPU_E, OPK_F] \leftarrow m_2.read();$
19 : $check(EPU_E);$
20 : $K_S \leftarrow X3DH(EPU_E, OPK_F);$
```

```
21 : $data \leftarrow collectRawData();$
22 : $N_1 \leftarrow genNonce();$
23 : $m_3 \leftarrow [data, N_1];$
24 : $E_{E2} \leftarrow Enc(K_S, m_3);$
25 : $E_{E2}.sendTo([ID_F]);$
26 :                   $E_{E2} \leftarrow receiveMsg();$
27 :                   $m_3 \leftarrow Dec(K_S, E_{E2});$
28 :                   $[data, N_1] \leftarrow m_3.read();$
29 :                   $check(N_1);$
30 :                   $N_2 \leftarrow genNonce();$
31 :                   $[ID_A, \Delta t] \leftarrow HAR(data);$
32 :                   $m_4 \leftarrow [ID_A, \Delta t]);$
33 :                   $E_{F2} \leftarrow Enc(K_S, [m_4, N_2]);$
34 :                   $E_{F2}.sendTo([ID_E]);$
35 : $E_{F2} \leftarrow receiveMsg();$
36 : $m_4 \leftarrow Dec(PR_E, E_{F2});$
37 : $[ID_A, \Delta t, N_2] \leftarrow m_4.read();$
38 : $check(N_2);$
39 : $N_3 \leftarrow genNonce();$
40 : $feedback \leftarrow validate(ID_A);$
```

```
41 : $m_5 \leftarrow [feedback, N_3];$
42 : $E_{E3} \leftarrow Enc(K_S, m_5);$
43 : $E_{E3}.sendTo([ID_F]);$
44 :                   $E_{E3} \leftarrow receiveMsg();$
45 :                   $m_5 \leftarrow Dec(K_S, E_{E3});$
46 :                   $[feedback, N_3] \leftarrow m_5.read());$
47 :                   $check(N_3);$
48 :                   $E_{F3} \leftarrow [ID_A, data, feedback];$
49 :                   $E_{F3}.sendTo([ID_C]);$
50 :                                  $E_{F3} \leftarrow receiveMsg();$
51 :                                  $[ID_A, data, feedback] \leftarrow E_{F3}.read();$
52 :                                  $store();$
```

Figure 3.3: Messages exchanged during (a) X3DHKA, (b) Activity Recognition, and (c) Cloud Update.

is possible to generate K_S, while the digital signature Sig_{EPU} guarantees that EPU_E is authentic.

After receiving E_{E1}, the fog checks Sig_{EPU} and computes the session key K_S using the X3DHKA scheme. Now only the fog device knows K_S; thus, in order to let the edge device to compute the session key, the message E_{F1} is sent (line 15). This message specifies both the one-time prekey OPK_F used by the fog device, and the $\mathsf{EPU_E}$ previously sent by the edge device so as to ensure the uniqueness of the session.

When the edge device receives $\mathsf{E_{F1}}$, it verifies that $\mathsf{EPU_E}$ corresponds to the one reported in message $\mathsf{E_{E1}}$ and performs all the steps needed to obtain $\mathsf{K_S}$ (lines 16-20). From now on, the two entities can delete all the intermediate keys and communicate through an Authenticated Encryption with Associated Data (AEAD) algorithm.

Concluded the key exchange steps, the edge device can start sending to the fog device the sensory data collected during the HAR phases (Figure 3.3(b), line 25). The field data includes both the feature vector computed on the accelerometer and gyroscope measures, and the $\mathsf{timestamp}$ reporting when data were captured. Moreover, in order to face a replay attack within the established session between edge and fog, $\mathsf{E_{E2}}$ also contains a nonce $\mathsf{N_1}$.

The messages of type $\mathsf{E_{E2}}$ are collected by the fog until it has enough rough data to start the activity recognition process. Once an activity has been recognized, the fog device prepares the message $\mathsf{E_{F2}}$ containing the $\mathsf{ID_A}$ of the activity, the time interval within it was performed, and a nonce $\mathsf{N_2}$ ensuring the uniqueness of the message (line 34). Finally, the message $\mathsf{E_{F3}}$ including all data involved in the activity recognition process is transferred from the fog to the cloud through a secure TLS channel (line 49), and then stored (line 52).

It is worth noting that X3DH Key Agreement (Figure 3.3(a)) is common to every kind of application in which SMCP is adopted. On the contrary, different crowdsensing applications may require the protection (Figure 3.3(b)) and synchronization (Figure 3.3(c)) of different kind of data. To this aim, the overall structure of SMCP can be easily modified and just a few message contents need to be changed according to the algorithms that implement the crowdsensing routines. For instance, considering a distributed vehicle traffic monitoring system, message $\mathsf{m_3}$ should be modified to contain other sensory data, such as accelerometer or GPS information. Then, many $\mathsf{m_3}$ messages would be sent to the fog where an ad-hoc machine learning algorithm (line 23) would be able to recognize the traffic level in a certain geographical area. Finally, the system could ask interested users to provide feedback to validate the reported output (line 40).

Chapter 4

Experimental Assessment

All the solutions described so far are evaluated in this Chapter. Section 4.1 briefly introduces the way the data were collected, and the devices used during the experiments. Then, Section 4.2 discusses a preliminary analysis about the ability of smartphones in recognizing simple activities with the aim to investigate and understand which are the parameters necessary for the HAR process. Finally, the Section 4.3 describes an exhaustive evaluation of both the distributed HAR framework and the secure protocol.

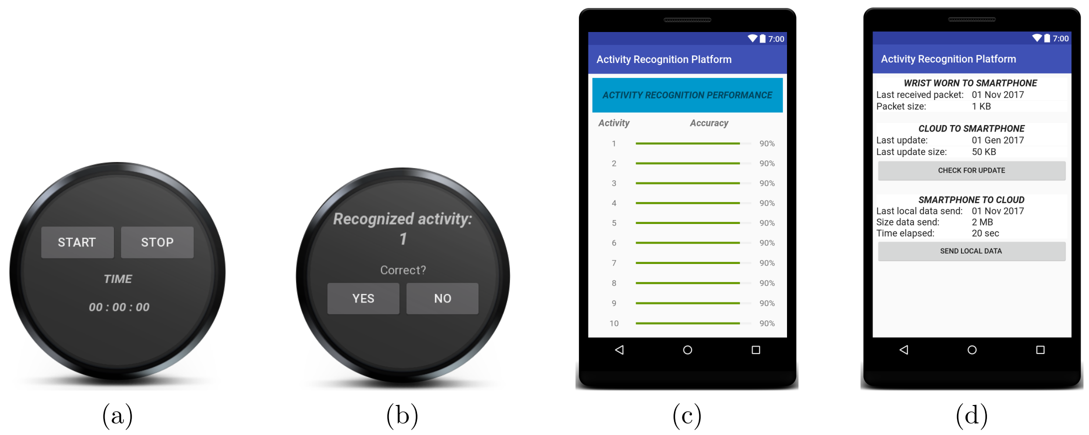
4.1 Experimental Setup

The experiments were carried out using three different models of Android-based smartphones and three smartwatches equipped with built-in accelerometer and gyroscope sensors (on the left side of Table 4.1). Two Android applications (one per device type) were developed to perform activity recognition and some supporting tasks, such as data management and labeling, compression, and secure transmission. The smartphone application can be installed on any Android device with Ice Cream Sandwich OS or higher, while the smartwatches require at least Jelly Bean OS.

Figure 4.1 shows four different screens of the Android application developed for the distributed HAR framework. The two leftmost images represent the smart-

	Edge			**Fog**			**Cloud**
	ED_1	ED_2	ED_3	FD_1	FD_2	FD_3	CDC
Type	*Smartwatch*	*Smartwatch*	*Smartwatch*	*Smartphone*	*Smartphone*	*Laptop*	*commercial*
CPU (GHz)	1.1	1.2	1.0	1.8	2.0	3.9	*cloud*
# core	4	4	2	8	8	4	*computing*
RAM (MB)	512	512	768	3072	4096	8192	*web service*

Table 4.1: Smart devices used in the proposed case study.



7:00
Activity Recognition Platform
ACTIVITY RECOGNITION PERFORMANCE

Activity	Accuracy
1	90%
2	90%
3	90%
4	90%
5	90%
6	90%
7	90%
8	90%
9	90%
10	90%

7:00
Activity Recognition Platform

WRIST WORN TO SMARTPHONE

Last received packet:	01 Nov 2017
Packet size:	1 KB

CLOUD TO SMARTPHONE

Last update:	01 Gen 2017
Last update size:	50 KB

CHECK FOR UPDATE

SMARTPHONE TO CLOUD

Last local data send:	01 Nov 2017
Size data send:	2 MB
Time elapsed:	20 sec

SEND LOCAL DATA

Figure 4.1: Smartwatch and smartphone Android applications. (a) The overall activity recognition process starts by pressing the *start* button on the wrist-worn interface. (b) Data are processed by the smartphone and the user is asked about the correctness of the classification. Detailed information about the recognition results (c), and data transmission statistics (d) can be examined trough the smartphone-side app.

watch side of the app allowing users to start/stop the activity recognition process, monitor the activity duration (Figure 4.1a), and give feedback about the recognition correctness (Figure 4.1b). The other two images show the smartphone side of the app that permits the user to have in-depth information about the accuracy of the recognition process (Figure 4.1c) and the transmission of collected data (Figure 4.1d).

Two different datasets were used in the experiments, and they were collected by asking 20 volunteers to perform activities in a period of 3 weeks. These datasets were obtained using the accelerometer and gyroscope sensors at a sampling rate of 100 samples per second; whereas they differ in the devices used during data collection, the activities recorded, and the number of samples. Specifically, the first dataset is adopted for the preliminary analysis and contains, for each activity, features vectors extracted from the sensors that record user's movements while the smartphone in his pants pocket. Users were asked to record 10 min for single activity resulting in approximately 60k readings (for a single axis) for both the accelerometer and gyroscope. The analysis of these readings in fixed-time windows of 3 sec led to the extraction of a set of 200 features for each activity, resulting in a dataset composed of 800 samples per user. Then, a total of about 16k samples was obtained.

The second dataset was used to test the performance of the distributed HAR framework and the activities are described by accelerometer and gyroscope data while the user wears the smartwatch on his wrist. Using an approach very similar to the one described above, a dataset consisting of more than 30k samples was constructed. Unlike the first dataset, a pre-processing phase was conducted in which some samples, considered redundant with respect to the activity performed, were removed in order to balance the dataset.

To collect data in a natural manner, in both cases the users were not provided with particular instructions on how to perform the activities, while they were simply informed about the activities to track and their meanings. Finally, users were asked to indicate the label corresponding to the activity performed.

4.2 Preliminary Analysis: Simple Activities

The experiments described here are carried out on three models of smartphones, as summarized in Table 4.1, in which a similar app to the one depicted in Figure 4.1 was installed.

The goal of this section is to investigate how difficult is recognizing activities using smartphones and discuss the related performance. To this aim, four simple activities were considered (i.e. *still*, *walking*, *running*, and *vehicle*), and it was conducted a comparison between the results obtained while using two ML algorithms, i.e. the k-nearest neighbors (K-NN) and a classifier based on k-means clustering. Then, the best of the two algorithm is chosen to be adopted in the sample HAR system described in Section 2.2.1.

The sample system is finally compared with two other well-known techniques to assess its effectiveness with respect to the state-of-art.

4.2.1 Choosing the classification algorithm

In the considered scenario, given the set of feature vectors $(f_1, f_2, \cdots, f_m)$, the most common application of the K-means algorithm consists in partitioning the m observations into k sets, $\mathbf{C} = (C_1, C_2, \cdots, C_k)$, so as to minimize the intra-cluster error:

$$E = \sum_{k=1}^{K} \sum_{f_i \in C_k} \|f_i - \mu_k\|^2 , \tag{4.1}$$

where μ_k is the mean value of the k-th cluster C_k.

Nevertheless, k-means can also be used for classification, i.e., supervised learning, according to two different schemes [Hastie et al. (2009)]. The first, straightforward, solution is to apply k-means to the whole blended training set and observe how data from k different classes are associated with each of the k centroids C_k. Then, each centroid is marked as representative of a certain class i, with $i = 1, ..., k$, if most of the samples in the cluster associated with C_k belong to i. Classification of a new, unknown, feature vector f is performed by finding its closest centroid and then assigning to f the same class of C_k. The major drawback to this method is that performing k-means on blended data produces heterogeneous clusters. i.e., there is no guarantee that all the points in the same cluster represent the same class.

The second approach helps to overcome this limitation by separating the training data in n distinct groups, each containing samples from one of the n classes to recognize. The k-means algorithm is applied on each group/class separately, so as to obtain k homogeneous clusters, i.e., all the centroids within a single group represents the same class. Thus, classification can be performed by comparing a new, unknown, feature vector f with the $k \times n$ labeled centroids, and assigning to f the class of the closest one.

The differences between the two methods are summarized in Figure 4.2. Original samples from three classes are represented by red, green, and blue points.

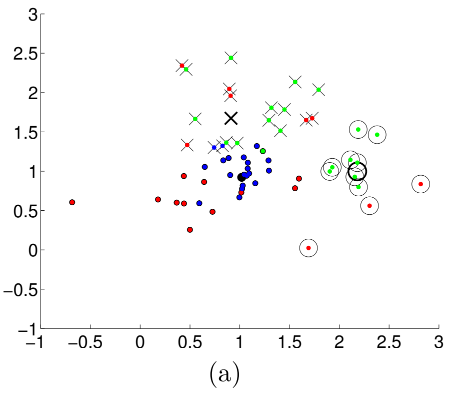


Figure 4.2: Examples of k-means classification adopting two different schemes. Data from three classes (red, green, and blue) are classified into three clusters (crosses, circles, and dots) using blended (a) or separated (b) training sets.

As a result of the classification, points are marked as belonging to one of three clusters denoted by crosses, circles, and dots. When using the first scheme, k-means is applied to the whole blended dataset (Figure 4.2(a)) so that each cluster contains data from different classes, e.g., the cluster denoted by crosses includes red, green, and blue points. On the contrary, the second scheme (Figure 4.2(b)) is preferable since it allows to apply k-means on each class separately, so obtaining homogeneous clusters, e.g., the elements of the cluster denoted by circles are all greens.

As regards the choice of k, some experiments were conducted to determine the best value for k-means and K-NN.

For k-means, the value of k is simply the number of activities to be recognized, i.e., $k = 4$. In order to find the best value of k for the K-NN algorithm, two techniques for predictive accuracy evaluation have been used, i.e., *re-substitution* and *cross validation*. Figure 4.3 shows the results of the experiments, with k ranging from 1 to 50. Considering that the classification algorithm is executed on devices with limited computational and energy resources, higher values of k have not been considered, since the computational complexity of the K-NN algorithm increases as k grows. Figure 4.3 shows that different values of k could allow the system to achieve an accuracy that is appropriate for the considered application, i.e., 4, 7, and 28 to 34. For these values, in fact, a good compromise between

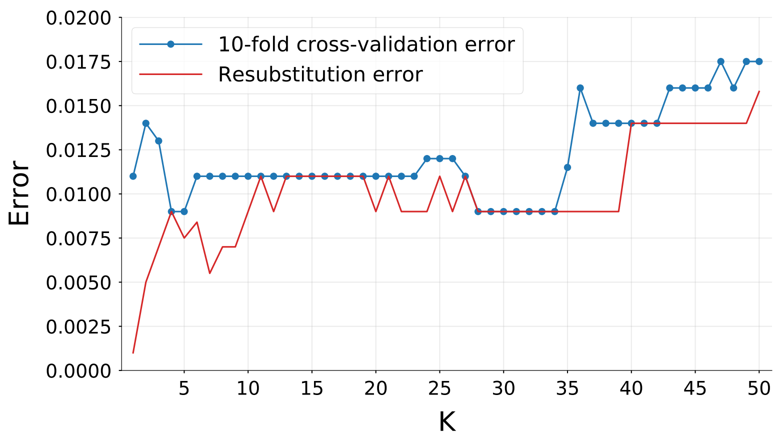
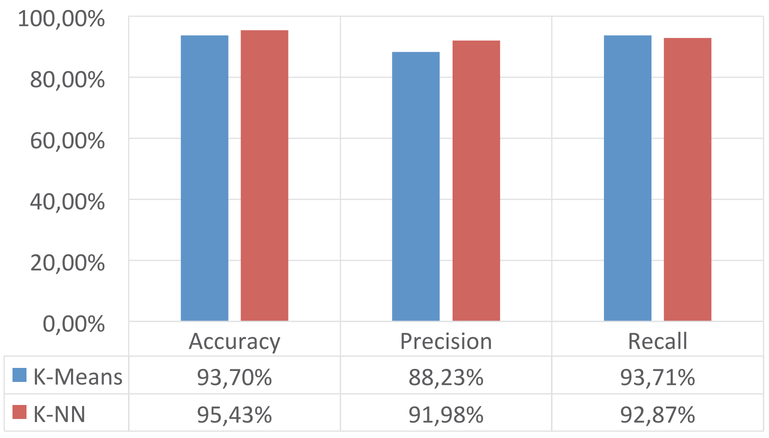


Figure 4.3: Error rate obtained with 10-fold cross validation and re-substitution with k ranging from 1 to 50.

the two metrics considered is obtained. In the final choice the values in the range [28-34] were discarded as they cause a higher complexity of the KNN algorithm; then, a value of k equal to 7 is used because it allows the system to achieve the lowest re-substitution error, while slightly increasing (about 0.0025) the 10-fold cross validation error. Obviously, this is a trade-off between accuracy and computational resources. Such trade-off can be evaluated differently depending on the particular application scenario, giving for example higher priority to energy saving or application responsiveness.

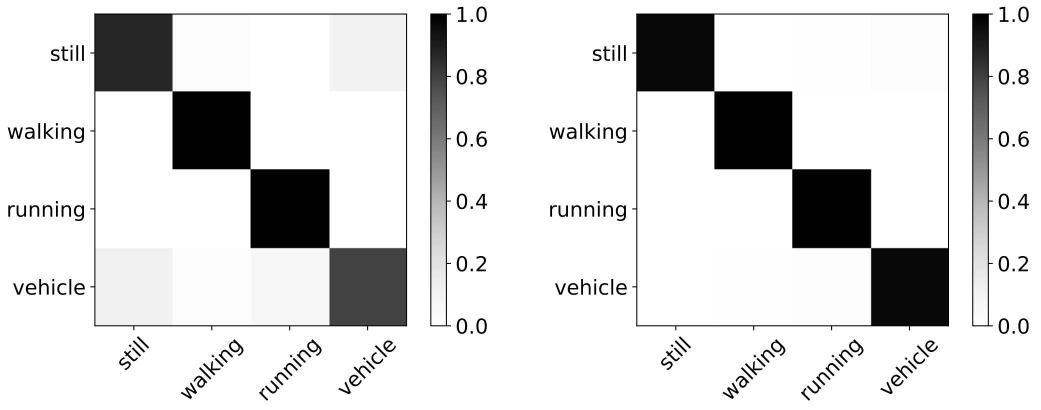
Having identified the k value, tests were performed to compare K-NN with k-means (scheme 2) in terms of *accuracy*, *precision*, and *recall* [Sokolova and Lapalme (2009)]. Figure 4.4 shows that slightly better results are obtained while applying the K-NN algorithm. This is mainly due to the incapacity of k-means to distinguish between some similar activities. In particular, as highlighted by the confusion matrices shown in Figure 4.5a and Figure 4.5b, *still* and *vehicle* activities are frequently mistaken because of their similar acceleration values (see Figure 2.2). This error is almost negligible when adopting the K-NN classifier.

The next set of experiments were aimed at comparing k-means and K-NN in terms of time of execution and memory consumption. Some tests were performed to measure these two parameters while varying the duration of the processing windows. Results are shown in Figure 4.6(a) and Figure 4.6(b). For the first



	Accuracy	Precision	Recall
🟦 K-Means	93,70%	88,23%	93,71%
🟥 K-NN	95,43%	91,98%	92,87%

Figure 4.4: Accuracy, precision, and recall of k-means and K-NN.



(a) (b)

Figure 4.5: k-means confusion matrix, and (b) K-NN confusion matrix.

two cases, i.e., test A and test B, the duration of the window is about 2 minutes, whereas smaller windows of about 50 seconds where used for C, D, and E. k-means is generally faster than K-NN, whilst K-NN, independently of the length of time windows, requires almost constant memory consumption.

Thus, since the HAR application must be as lightweight as possible so as to prevent the system resources from being consumed more than necessary, it was decided to build the classification module on the K-NN algorithm. Such analysis is also confirmed by the results reported in [Munther et al. (2016)].

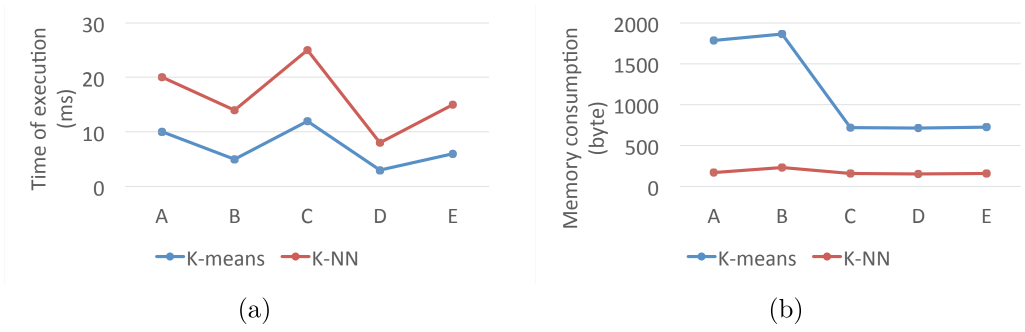


Figure 4.6: Time of execution (a) and memory consumption (b) of K-means and K-NN classifiers measured under five different conditions denoted by {A,B,C,D,E}.

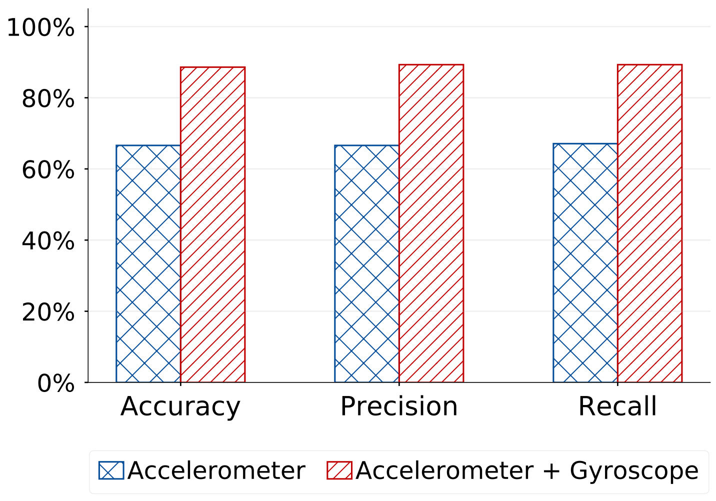
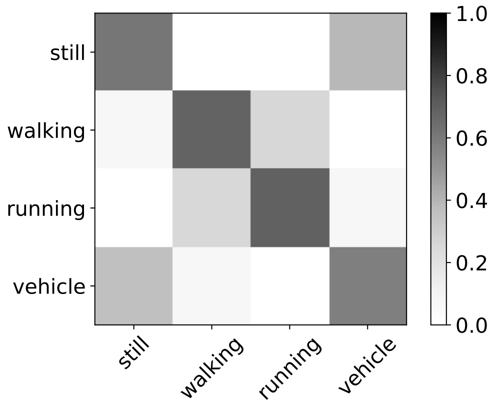


Figure 4.7: Accuracy, precision, and recall obtained by exploiting only accelerometer data, or both accelerometer and gyroscope data.

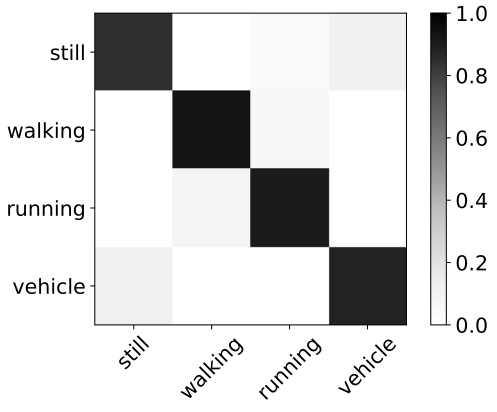
4.2.2 An Early Discussion on the Motion Sensors

To underline the importance of merging data coming from heterogeneous sensors, let's consider two different systems that use the K-NN algorithm, as described in the previous section. The first one uses only acceleration values, while the second leverages on data coming from both accelerometer and gyroscope.

Figure 4.7 shows accuracy, precision, and recall obtained by both systems. As expected, results confirm that gyroscope data are extremely important.



(a)



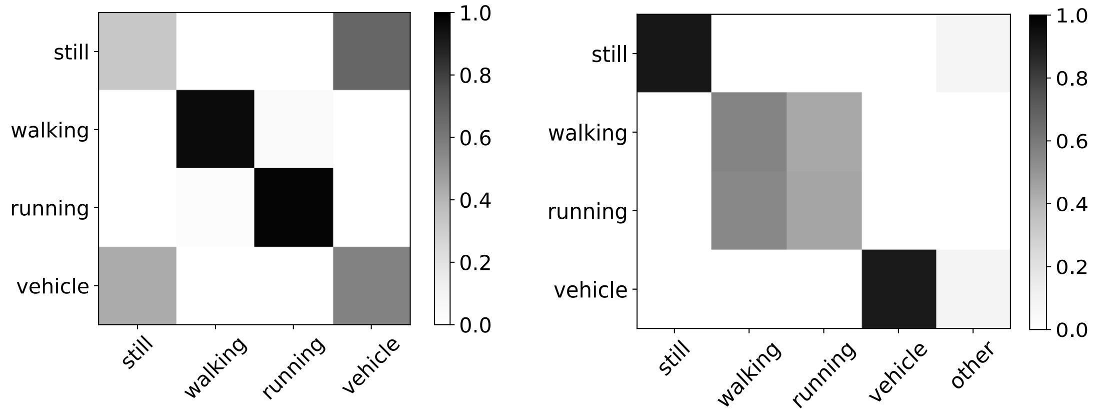
(b)

Figure 4.8: Confusion matrix obtained by considering only accelerometer sensor data, and (b) confusion matrix obtained by considering both accelerometer and gyroscope sensor data.

In fact, exploiting only accelerometer data yielded an accuracy of about 65%, compared to about 90% obtained by fusing data from both sensors. Therefore, the experimental evidence concludes that it is extremely advantageous to exploit data coming from the gyroscope, in addition to those of the accelerometer.

To better understand the reasons for this marked difference, as well as to analyze more deeply the results obtained, in Figure 4.8, the confusion matrices obtained by the two systems are presented.

Figure 4.8a shows that the system exploiting only accelerometer data has difficulty in correctly classifying the activities of *still* and being in a *vehicle*. Also, even *running* and *walking* are often confused with each other. These difficulties can be easily explained by the fact that data come from only one type of sensor, which cannot unambiguously describe the patterns of some activities. In fact, using only the accelerometer it is difficult to distinguish between a person who is simply stationary or stationary in a vehicle. As expected, adding gyroscope data overcomes the problem. The confusion matrix in Figure 4.8b has a very marked main diagonal, which shows how the system is able to recognize all activities satisfactorily, without confusing them with each other.



(a) (b)

Figure 4.9: (a) MosT confusion matrix, and (b) Google confusion matrix.

4.2.3 Performance Assessment of the Sample HAR System

In order to assess the system's performance, this section present a comparison with two state-of-art HAR techniques, i.e., those implemented by the MosT framework [Cardone et al. (2016, 2013)] and by the Google APIs [Google (2016)].

As mentioned previously, Google does not provide any detail of the algorithms behind their products, thus their recognition algorithm was treated as a black-box. On the contrary, MosT is based on a well known algorithm to efficiently build decision trees, namely C4.5 [Quinlan (2014)].

Since MosT and Google are able to recognize different types of activities, in order to perform a meaningful assessment two distinct subsets have been defined. More precisely, the class *other* was added to cover the activities not considered in both of the systems alternately compared. Thus, since Google technique is unmodifiable, and it recognizes a greater number of activities than the sample system, the comparison was based on a subset formed by *still*, *walking*, *running*, *vehicle*, and *other*. On the contrary, even if MosT originally included only *still*, *walking*, and *running*, the *vehicle* activity was added obtaining the same set addressed by the sample HAR system.

Accuracy, precision, and recall achieved by the proposed system, as compared to MosT, are showed in Figure 4.10. MosT results are detailed in the confusion matrix reported in Figure 4.9a. As expected, *walking* and *running* are almost

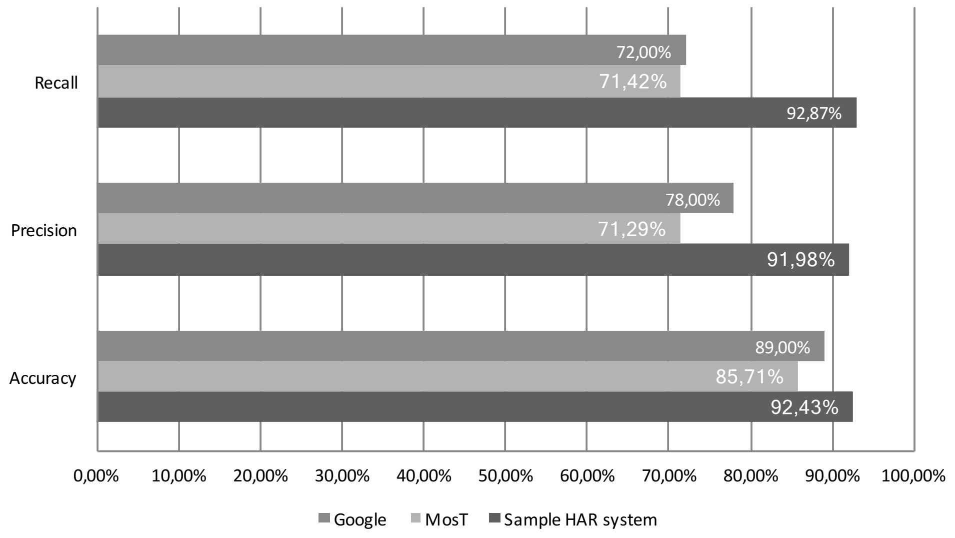


Figure 4.10: Comparison between the proposed system, MosT, and Google.

correctly classified, whilst the recognition of *vehicle* and *still* is more difficult to perform. This can be explained because MosT considers only accelerometer data, that, as shown in Figure 2.2, are not useful enough for discriminating between a still user and one driving at constant velocity. Moreover, decision trees are generally less predictive than other classification approaches, since a small change in the data can cause a large change in the final estimated tree [James et al. (2014)].

As regards the results obtained comparing the proposed system with the Google activity recognition tool (see Figure 4.10), it can be noticed that Google performances are quite lower than ours, and this can be explained by referring to the Figure 4.9b. In fact, the implementation provided by Google is not able to correctly distinguish between *walking* and *running* activities. In addition, as already discussed, it is not possible to run further experiments, similar to those described for MosT, to analyze how changing the set of activities would impact on the system performance. This represents a further advantages of the proposed system compared with Google method.

ID	Activity Name	Reference Scenario
CA_1	Go To Work 1	The user goes to work by walking for a while.
CA_2	Shopping	The user alternates still and walking phases.
CA_3	Relax	The user is sitting for a long time.
CA_4	Eating	The user is sitting and moves the hands up and down while eating.
CA_5	Working at PC	The user is sitting and types on the PC keyboard or uses the mouse.
CA_6	Cooking	The user is cooking briefly moving in the kitchen.
CA_7	Jogging	The user alternates running and walking phases.
CA_8	Go To Supermarket	The user goes to the supermarket alternating vehicle, walking, and still micro-activities.
CA_9	Go To Work 2	The user goes to work alternating walking and vehicle micro-activities.
CA_{10}	Driving	The user stays in a vehicle for a long time.

Table 4.2: Complex Activities Analyzed During the Activity Recognition Process.

4.3 Distributed HAR: Complex Activities

In order to assess the validity and generality of algorithms and the architecture described in Section2.2.2 and in Section 2.3, respectively, it was considered a scenario in which people acting in a smart environment, e.g., a smart city, a smart campus, or even a gym or a retirement home, are monitored through a pervasive artificial intelligence system whose nodes are the users' personal smart devices.

The effectiveness of the proposed HAR technique is proved by performing three different set of experiments tested on the dataset composed of the complex activities described in Table 4.2. The first aimed to find the best values (Ω, N) in terms of system accuracy and the F-score metric. The second was focused on understanding how the number of observed samples affects the performance of the activity recognition technique. Finally, it was investigated the impact of data exchange between the several entities involved in the HAR on the overall efficiency of the system. In this sense, data between smartwatches and smartphones are exchanged through short-range Bluetooth technology, so as to make the proposed architecture compatible with a number of smart devices that do not provide other wireless communication interfaces, such as 802.15.6 and ultra-low power WiFi.

Finally, since the HAR algorithms highly exploits the computational capabilities of resource-constrained devices, other experiments have been focused on evaluating the overhead introduced by the secure message exchange protocol and comparing the protocol with other approaches representing the state-of-the-art.

4.3.1 Activity Recognition Results

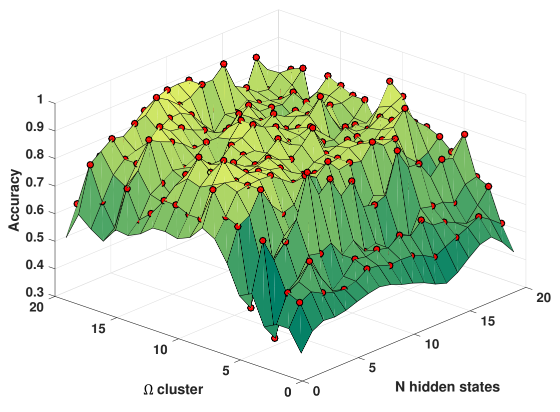
The first group of experiments aimed at finding the best set of parameters for the activity recognition procedure-that is, the best pair (Ω, N), where Ω is the number of clusters/words in the dictionary and N represents the number of hidden states in HMMs.

To this purpose, a grid-search approach [Bergstra and Bengio (2012)] was applied to measure the system performance in terms of accuracy, precision, recall, and F-score values [Davis and Goadrich (2006); Powers (2011)]. Generally, grid search is run with a cross-validation technique, such as K fold cross validation (KFCV) [Rodriguez et al. (2010)] or leave-one-out cross validation (LOOCV) [Arlot et al. (2010)]. The basic idea of KFCV is to partition a dataset into K folds to obtain a more realistic assessment of the considered model. Each time, one of the K subsets is used for testing and the other $(K - 1)$ for training. The average error across all iterations provides an estimation of the overall system performance. LOOCV is a special case of KFCV in which the number of folds is equal to the number of points in the dataset.

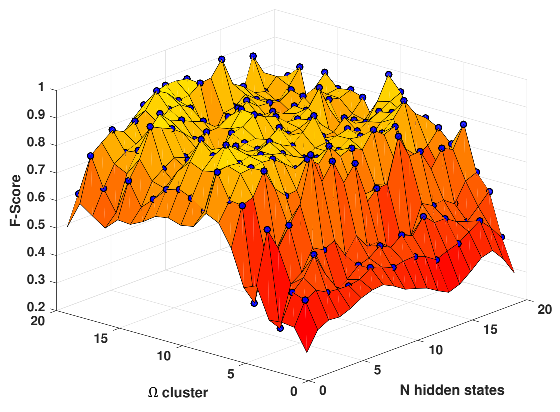
During the experimental evaluation, it is adopted a grid search on Ω and N guided by a KFCV to find the pair that provides the best accuracy and F-score values. The number of folds has been set to 10 so as to minimize the bias (i.e., the difference between estimated and actual accuracy) [Rodriguez et al. (2010)].

Figure 4.11 shows the results obtained for different iterations of the grid search algorithm on a training set S_1. Since considering accuracy values only (Figure 4.11a) can cause misleading evaluations, the F-score was also computed. Figure 4.11b shows that the best value of the F-score is obtained for $\Omega = 19$ and $N = 14$, that represent the optimal number of words and hidden states to use during the recognition phase. Detailed results of the KFCV for the considered set of complex activities CA are summarized in Table 4.3.

Once the best pair (Ω, N) has been found, the next set of experiments aimed to evaluate the capability of the system to recognize an activity from an unseen test



(a)



(b)

Figure 4.11: K-fold cross validation. Accuracy (a) and F-score (b) varying the number of clusters $\Omega \in [2, 20]$ and the number of hidden states $N \in [2, 20]$.

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	1	0	0	0	0	0	0	0	0	0
CA_2	0	0.6	0	0	0	0	0	0	0	0.4
CA_3	0	0	1	0	0	0	0	0	0	0
CA_4	0	0	0	1	0	0	0	0	0	0
CA_5	0	0	0	0	1	0	0	0	0	0
CA_6	0	0	0	0	0	1	0	0	0	0
CA_7	0	0	0	0	0	0	1	0	0	0
CA_8	0	0.2	0	0	0	0	0	0.8	0	0
CA_9	0	0	0	0	0	0	0	0	0.8	0.2
CA_{10}	0	0	0	0	0	0	0	0	0	1

Table 4.3: K-Fold Cross Validation confusion matrix for $\Omega = 19$ and $N = 14$.

set S_2, made of different repetitions of the complex activities listed in Table 4.2. Firstly, 10-fold cross validation on the new test set was performed, and the relative confusion matrix is showed in Table 4.4. Results show an average accuracy of 78% and an F-score value of 0.72. The confusion matrix also highlights that most of the recognition errors depend on the complex activities CA_8 , CA_9 , and CA_{10}.

A further set of experiments has been performed to measure the system performances while considering different training sets obtained by randomly choosing samples from the set S_2. In experiment A, it was selected 1/3 of the samples to train the system and the remaining 2/3 for testing; in experiment B, 2/3 of the samples were chosen to train the system and the remaining 1/3 for testing; and in experiment C, half of the samples were used for training and half for testing.

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.92	0	0	0	0	0	0	0	.08	0
CA_2	0	.8	0	0	0	0	0	.06	0	.14
CA_3	0	0	.86	0	.14	0	0	0	0	0
CA_4	0	0	0	.78	0	.1	.12	0	0	0
CA_5	0	0	0	0	.94	.06	0	0	0	0
CA_6	0	0	0	.12	0	.76	.12	0	0	0
CA_7	0	0	0	.04	0	.22	.74	0	0	0
CA_8	0	.22	0	0	0	0	0	.64	0	.14
CA_9	.14	0	0	0	0	0	0	.04	.66	.16
CA_{10}	0	.04	0	0	0	0	.08	.18	0	.7

Table 4.4: Ten-Fold Cross-Validation Confusion Matrix (Accuracy) for the Testing Phase.

Confusion matrices for each experiment are presented in Tables 4.5, 4.6, and 4.7, respectively. The worst performances are obtained for experiment A, in which the mean accuracy is equal to 71% and the F-score to 0.69. Better results can be obtained by increasing the number of training samples as showed in experiment B and experiment C. In particular, it can be noticed that when the considered training set is 2/3 of the original set (Table 4.6), the mean accuracy and F-score are comparable to the values obtained when considering the whole set (i.e., 88%, and 0.9 respectively). Similar considerations can be made for experiment C, in which accuracy is equal to 82% (Table 4.7) and the F-score to 0.79.

This set of experiments revealed that the system performances get worse when S_2 is reduced by a factor of three, while the HAR algorithm still provides a good recognition rate when the original dataset is reduced by 2/3 or 1/2. These last two results underline that the proposed system is able to capture a general model of the activity set.

4.3.2 Analysis of the HAR Dataflow

To discuss the computational effort required only for the recognition of the activities, this section investigates how data processing and transmission impact the distributed framework, given that as the amount of data to be managed increases, the smart device's battery life generally reduces [Ickin et al. (2012)]. Specifically, the focus is on the dataflow depicted in Figure 2.7, while a detailed analysis of the

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.79	0	0	0	0	0	0	0	.12	.09
CA_2	0	.58	0	0	0	0	0	.18	0	.24
CA_3	0	0	.88	0	.12	0	0	0	0	0
CA_4	0	0	0	.74	0	.03	.24	0	0	0
CA_5	0	0	0	0	.97	0	.03	0	0	0
CA_6	0	0	0	0	0	.88	.12	0	0	0
CA_7	0	0	0	0	0	.24	.76	0	0	0
CA_8	0	.09	0	0	0	0	0	.48	0	.42
CA_9	.09	0	0	0	0	0	0	.09	.55	.27
CA_{10}	0	.24	0	0	0	0	.09	.21	0	.45

Table 4.5: Confusion matrix for experiment A: 1/3 training set, and 2/3 test set.

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.88	0	0	0	0	0	0	.06	0	.06
CA_2	0	.82	0	0	0	0	0	.18	0	0
CA_3	0	0	1	0	0	0	0	0	0	0
CA_4	0	0	0	.94	0	.06	0	0	0	0
CA_5	0	0	0	.06	.94	0	0	0	0	0
CA_6	0	0	0	0	0	.88	.12	0	0	0
CA_7	0	0	0	0	0	0	1	0	0	0
CA_8	0	.12	0	0	0	0	0	.76	0	.12
CA_9	0	0	0	0	0	0	0	0	.83	.17
CA_{10}	0	0	0	0	0	0	0	.18	0	.82

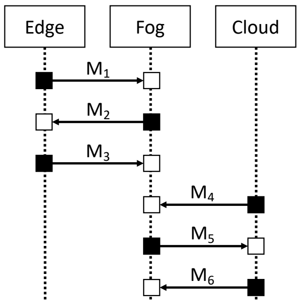
Table 4.6: Confusion matrix for experiment B: 2/3 training set, and 1/3 test set.

	CA_1	CA_2	CA_3	CA_4	CA_5	CA_6	CA_7	CA_8	CA_9	CA_{10}
CA_1	.88	0	0	0	0	0	0	0	0	.12
CA_2	0	.72	0	0	0	0	0	.2	0	.8
CA_3	0	0	.92	0	.08	0	0	0	0	0
CA_4	0	0	0	.76	0	.08	0.16	0	0	0
CA_5	0	0	0	0	.96	0	0	0	.04	0
CA_6	0	.04	0	0	0	.96	0	0	0	0
CA_7	0	0	0	0	0	.08	.92	0	0	0
CA_8	0	.12	0	0	0	.08	0	.64	0	.16
CA_9	0	0	0	0	0	0	0	0	.76	.24
CA_{10}	0	.2	0	0	0	0	0	.12	0	.68

Table 4.7: Confusion matrix for experiment C: 1/2 training set, and 1/2 test set.

overhead introduced by the secure protocol is given in Section 4.4. For the sake of clarity, a summary of the HAR dataflow is showed in Fig. 4.12.

Some preliminary tests to evaluate the *wrist-worn to smartphone* data transmission were performed during the design of the system. Results showed that the impact of such a transmission on the overall performances is negligible. In particular, accelerometer and gyroscope data can be sent to the smartphone over a Bluetooth connection, with a transfer rate of 25Mbps (approximately 3.125MB/s).



Message	Description
M_1	It contains the feature vector.
M_2	It contains the recognized activity.
M_3	It contains the user feedback about the activity.
M_4	It is used to check if new data are available.
M_5	It contains all data buffered during the previous phase.
M_6	It contains the updated model for the recognition.

Figure 4.12: Summary of the distributed HAR dataflow discussed in Section 4.4.

Given that tracking an activity produces less than 100KB per minute, and data are transferred from wrist-worn to smartphone devices every 10 minutes, it is possible to conclude that the transmission of the first four messages can be performed timely without a significant impact on the battery life. However, the transmission of M_5, that contains a set of sensory data and user feedback collected after a particular activity is recognized, and M_6, created when the dictionary and the parameters of the HMMs are updated, could affect the performance of the system.

Results indicate that the average time needed to transmit the message M_6 using a WiFi connection is quite low, about 0.2 s, while the transmission of sensory data and user feedback require a noteworthy amount of time, about 7 s. To better investigate this aspect, other experiments were performed so as to determine the relationship between the duration of the data collection process and the size of M_5. The outcomes suggest that as collection time increases, then the size of data transmitted from clients to the cloud grows very rapidly. This is mainly due to the adoption of JSON-formatted messages which, despite speeding up the transmission compared to XML [Nurseitov et al. (2009)], include auxiliary text to generate and parse every pair attribute/value.

To deal with this aspect, the adoption of two lossless compression techniques (i.e., compressed JSON (CJSON) and GZIP [Bell et al. (1990)]) has been considered. The idea behind CJSON is to exploit some redundant information from the original JSON message to obtain a certain level of compression. The GZIP algorithm is a variation of the LZ77 data compression algorithm that includes Huffman coding. Lossless compression allows to significantly reduce the size of the data to

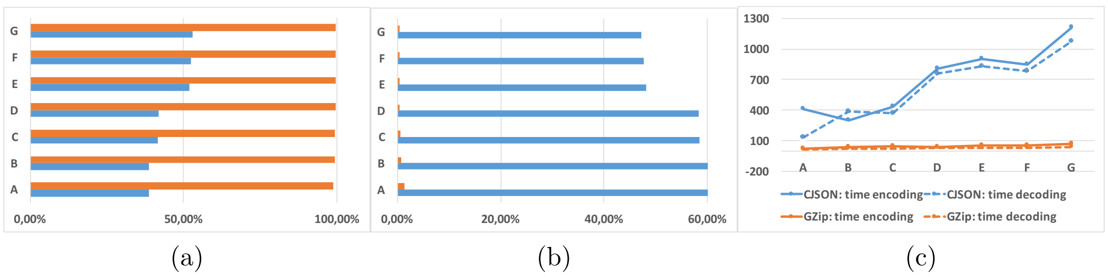


Figure 4.13: Average Saving Percentage (a), Compression Rate (b), and compression/decompression time (c) of CJSON (blue) and GZIP (orange) algorithms.

be transferred from the fog to the cloud without losing any information that may affect the performance of the HAR system.

Experiments aimed at comparing the effectiveness of the two algorithms in terms of saving percentage (SP) and compression ratio (CR) [Kodituwakku and Amarasinghe (2010)]. Moreover, to evaluate their suitability to the scenario discussed so far, compression and decompression time have also been computed. Assuming that S_M is the original size of the message and S_m the size after compression, then the saving percentage, SP, and the compression ratio, CR, can be computed as

$$SP(\%) = \frac{S_M - S_m}{S_M}, \quad CR = \frac{S_m}{S_M}. \tag{4.2}$$

Tests were run on smartphone devices compressing several messages of various sizes (from 500KB up to 4MB), and results are summarized in Figure 4.13. It is possible to observe that GZIP outperforms CJSON both in terms of saving percentage (Figure 4.13a) and compression rate (Figure 4.13b). Moreover, Figure 4.13c shows that GZIP compression/decompression times measured while increasing the input size (from A to G) are quite lower than those achieved by CJSON. Thus, to contrast the behavior observed in Figure 4.13b, GZIP compression of sensory data is performed to reduce bandwidth usage and enable faster communication between the smartphones and the cloud.

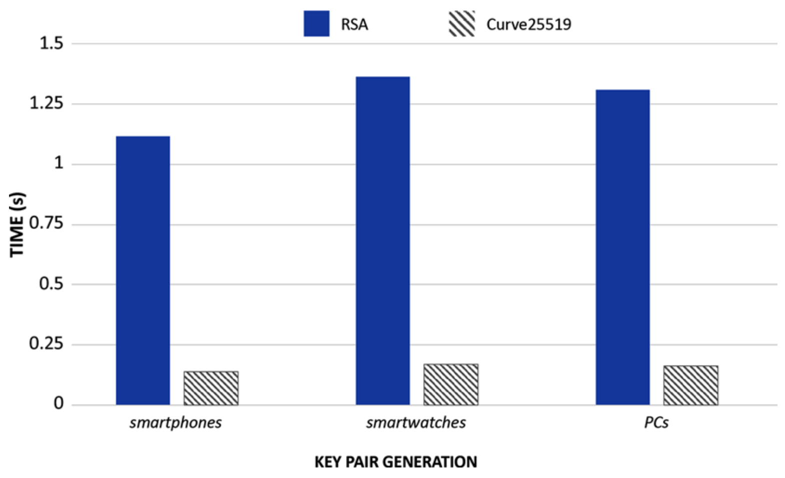


Figure 4.14: Average computation time required to generate the key pair by RSA and Curve25519 on smartphones, smartwatches, and PCs.

4.4 Distributed HAR Security

Since the HAR algorithms highly exploits the computational capabilities of resource-constrained devices, tests presented here have been focused on evaluating the overhead introduced by the secure message exchange protocol.

The first set of experiments aimed at measuring the computation time required by RSA and Curve25519 to generate a key pair on different edge and fog devices. Since Curve25519 uses keys of 256-bit guaranteeing a security level of 128-bit, comparisons with RSA were performed using keys of 3072-bits that provide roughly an equivalent resistance to security attacks.

Figure 4.14 shows the average computation time required to generate the key pair, as measured on smartwatches, smartphones, and PCs. Results confirm the effectiveness of elliptic-curve cryptography being each of the three tasks completed in about 0.15 s. Moreover, Curve25519 allows to use shorter keys than RSA, which also lead to better storage requirements and improved performance.

As regards data transmission, the devices operating within the proposed framework can transmit to each other information of different kind (e.g., sensor measurements, messages, activity models), and size. Then, other tests were performed to evaluate the complexity of the messages described in Section3.4.3 in terms of computation time required for their preparation t_p, and size s.

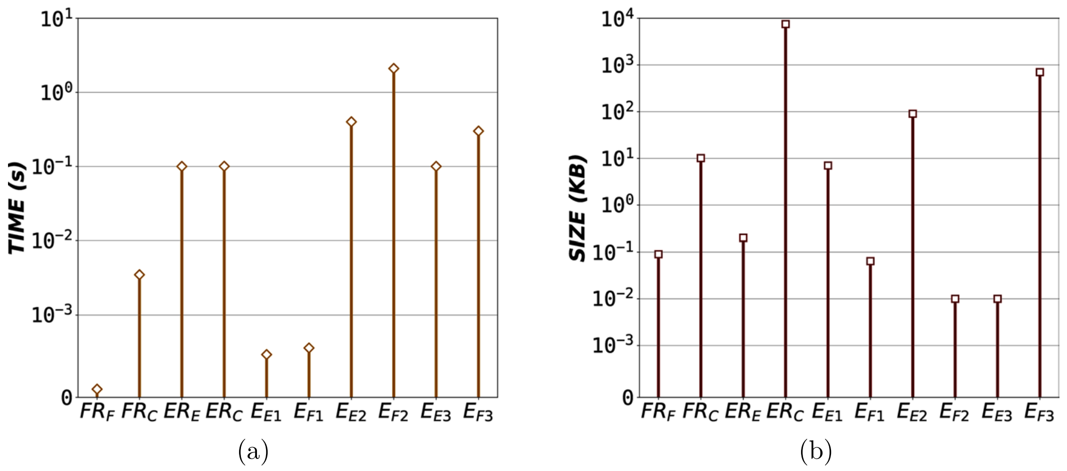
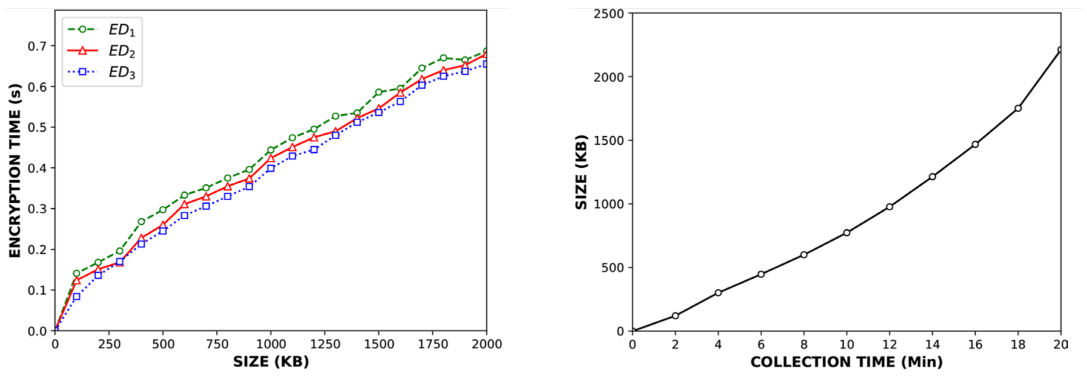


Figure 4.15: Message complexity in terms of computation time (a) and size (b).

Figure 4.15a shows that a time of about 2 s is required to compute the most complex message E_{F2}. This message is created by a fog device after the recognition process has been completed; thus, a low value of $t_p(\mathsf{E}_{\mathsf{F2}})$ reflects also the good performances of the HAR algorithms. Similarly, the second most time expensive messages E_{E2} and E_{F3}, containing recognition *data*, are prepared in about 0.4 s. The computation time required to prepare the messages ER_{E}, ER_{C}, and E_{E3} mostly depends on the symmetric encryption step. By observing the other t_p values, it is possible to conclude that the overhead introduced by the secure message exchange protocol is very low, being the average computation time for the other messages below 0.3 s.

The size of the set of messages discussed so far is analyzed in Figure 4.15b. The heaviest message, as expected, is the set ER_{C} that contains the fullEdgeApp sent to the edge relying on the fog. The second heaviest message is E_{F3}, that is the synchronization message sent from the fog to the cloud in order to update the overall activity models. The size of all the messages from FR_{F} to E_{F1} is always smaller than 10 KB, making them suitable for timely transmission also through the low-power edge-fog communication network.

As regards E_{E2} its size varies according to the amount of data transferred from the edge to the fog. It is worth noting that this message is encrypted with the



(a) (b)

Figure 4.16: (a) Size of E_{E2} while varying the data collection time, and (b) the corresponding average encryption time using AES.

session key K_S, thus the greater is its size the more is the computation time required by the symmetric- key encryption algorithm.

In order to find a trade-off between message size and encryption time, different tests were performed using the devices listed in Table 4.1. The curves depicted in Figure 4.16a show that the time required for encrypting the message E_{E2} is similar for the three adopted edge devices. This result is easily explainable since AES is efficiently performed in hardware in almost any recent smart device. According to these results, the size of E_{E2} was limited to about 100 KB so as to perform AES encryption in about 100 ms. Moreover, as shown in Figure 4.16b, such an amount of data is collected in about 2 min when using the same sampling frequency for all the devices.

4.4.1 Comparison with State-of-Art Protocols

In order to present a preliminary comparison of SMCP with other approaches representing the state-of-the-art, Table 4.8 summarizes the security features provided by some relevant related protocols. It is possible to notice that some of them, i.e., [Amor et al. (2017)] and [Ibrahim (2016)], do not cover secure fog-cloud communications, nor provide user anonymity. None of these protocols but SMCP are able to manage dynamic fog device registration, while all are designed to deal with replay attacks. According to these results, in the following of this section it will

Feature	**SMCP**	**Wazid et al. (2019)**	**Amor et al. (2017)**	**Ibrahim (2016)**	**Hu et al. (2017)**	**Wang et al. (2018)**
Dynamic fog device registration	✓					
Edge registration	✓	✓	✓	✓		✓
Fog registration	✓	✓		✓		
Secure edge-fog communication	✓	✓	✓	✓		✓
Secure fog-cloud communication	✓	✓			✓	✓
Fog device monopolization	✓					
Offline password guessing attack	✓	✓		✓		
Revocation policy		✓		✓		✓
User anonymity	✓	✓			✓	✓
Replay attack	✓	✓	✓	✓	✓	✓
User impersonification attack	✓	✓			✓	

Table 4.8: Security features comparison among SMCP and some other protocols.

be presented a comparative analysis of SMCP with a general-purpose protocol, namely SAKA [Wazid et al. (2019)], and a specific application domain protocol, FIRF [Hu et al. (2017)].

The main goal of the SAKA is to provide a secure authenticated key agreement scheme that is suitable for a general purpose fog-based application. The protocol consists of eight phases, among which it can be identified those that are strictly related to the four that characterize SMCP, namely the *fog registration*, *edge registration*, *edge-fog communication*, and *fog-cloud communication*.

The first set of experiments was focused on comparing SMCP and SAKA in terms of computation time spent for completing the four aforementioned phases. To this aim, it was followed the same approach adopted by the authors of SAKA. Starting from the sequence of messages used in SMCP, it was analyzed each message by pointing out the operations that mostly affect the computation time. The same analysis was performed on SAKA by defining a new sequence of messages (see Figure 4.17) that provide the same functionalities of SMCP while using the SAKA key agreement scheme. Then, it was selected the most heavy operations from both protocols, whose computation times are denoted by T_{ecm} (elliptic curve point multiplication), T_{hash} (cryptographic one-way hash function), T_{sig} (signature gen-

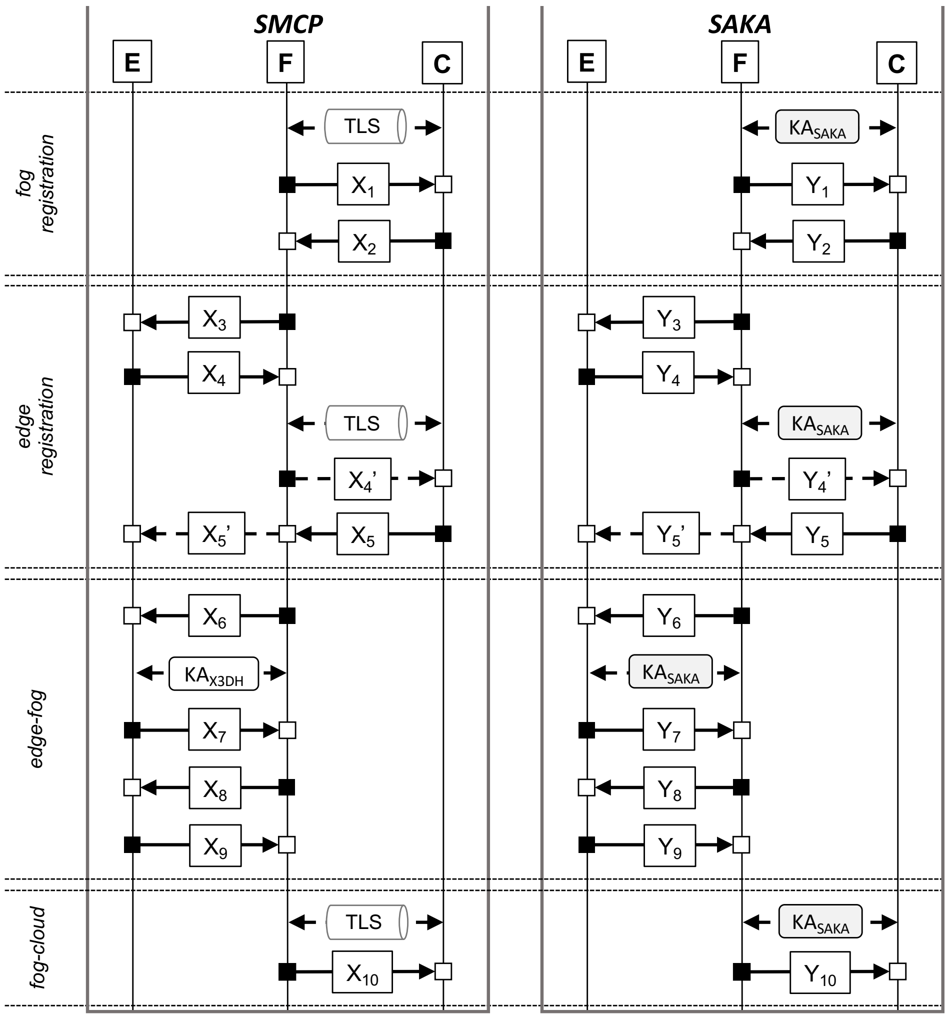


Figure 4.17: Messages exchanged during four phases common to the SMCP and SAKA protocols.

eration), T_{sed} (symmetric encryption and decryption), T_{aed} (asymmetric encryption and decryption), T_{tls} (TLS handshake), T_{cert} (certificate generation), T_{x3dh} (X3DHKA protocol), T_{kafc} (SAKA fog-cloud key agreement), and T_{kaef} (SAKA edge-fog key agreement).

These quantities have been used to provide an early, explainable, description of the computation cost of SMCP and SAKA. Results, summarized in Table 4.9, highlight that some computations are common to both protocols, i.e., symmetric encryption and decryption (*sed*) and signature generation (*sig*), while others reflect

	SMCP	**SAKA**
Fog Registration	$T_{tls} + 2T_{cert} + 2T_{sed}$	$T_{kafc} + 2T_{sed}$
Edge Registration	$3T_{sed} + T_{tls} + T_{sig} + T_{cert} + 2T_{ecm}$	$3T_{sed} + T_{kafc} + T_{sig}$
Edge-Fog	$T_{x3dh} + 4T_{sed}$	$T_{kaef} + 4T_{sed}$
Fog-Cloud	$T_{tls} + T_{sed}$	$T_{kafc} + T_{sed}$
Total cost	$3T_{tls} + 3T_{cert} + 9T_{sed} + T_{sig} + T_{x3dh}$	$3T_{kafc} + 9T_{sed} + T_{sig} + T_{kaef}$

Table 4.9: Comparison of SMCP and SAKA computation costs.

the main distinctive characteristic of the two schemes, i.e., the key agreement phase. More specifically, SMCP makes use of TLS to protect the communications between fog and cloud, and exploits the X3DHKA protocol for key agreement in edge-fog message exchange. On the other hand, SAKA proposes an ad-hoc key agreement protocol for making secure edge-fog and fog-cloud communications.

These preliminary results have been further investigated by running SMCP and SAKA on the devices presented at the beginning of this section. Results revealed that SMCP takes less computation time than SAKA to complete each phase. In particular, *fog registration*, *edge registration*, and *fog-cloud* communications are faster in the proposed scheme thanks to the adoption of the TLS protocol. As regards the *edge-fog* communication, the total computation time of SMCP is almost equal to the corresponding phase in SAKA.

Conversely, TLS impacts on the size of the messages because of the introduction of security certificates. Moreover, during the SMCP *fog registration* and *edge registration*, the transmission of other two certificates ($\mathsf{Cert_F}$ and $\mathsf{Cert_E}$) increases the message size. However, it is worth noting that certificates allow us to overcome three notable limits of SAKA. The first is that fog devices considered by SAKA are chosen during the design of the fog application; it means that the set of fog devices is fixed, and they are implicitly treated as trusted entities. On the contrary, SMCP allows to use a wide set of fog devices that can join the system at anytime by just presenting themselves by means of their certificates. Moreover, the use of security certificates allows to prove the characteristics of the device, preventing an internal attacker from manipulating the system during the selection of the *best* fog to provide a service.

Secure fog-cloud communications	**Time (ms)**	**Message size (KB)**
FIRF: Z_1	0.41	0.17
FIRF: Z_2	0.42	0.18
FIRF: Z_3	0.02	0.12
FIRF: Z_4	0.01	0.01
FIRF (total)	0.86	*0.48*
SMCP: TLS	*0.56*	1.50

Table 4.10: SMCP and FIRF compared in terms of average execution time (s) and size (KB) of the messages exchanged during the fog-cloud communication.

Finally, a notable drawback of SAKA is the huge network load generated when a new edge device is registered to the network. In particular, every time a new edge device is added, the cloud sends a message to all fog devices via a secure channel. This represents a significant limitation in real applications where the number of edge and fog devices is high.

The last set of experiments aimed at comparing the secure fog-cloud communications provided by SMCP and FIRF [Hu et al. (2017)]. In SMCP, every communication between fog and cloud is secured by TLS, while FIRF adopts a preliminary session key agreement protocol, based on Diffie-Helmann, that requires a set Z formed of 4 different messages. The first two, Z_1 and Z_2 , are used to obtain the session key, while Z_3 and Z_4 to check if the key exchange has been completed successfully. A comparison between different executions of the two protocols focusing on fog-cloud communications only is reported in Table 4.10. Results show that the use of TLS makes SMCP able to complete the message exchange faster than FIRF, while message size is still worst in SMCP due to the use of certificates. Furthermore, SMCP allows the system to achieve a higher degree of security compared to FIRF; for instance, data secrecy, data integrity, and users anonymity are not fully covered by FIRF due to the lack of secure communication between edge and fog devices, as discussed in Section 3.1.

Chapter 5

Conclusions and Future Works

In recent years, people's lifestyle has been totally revolutionized by the advent of devices whose capabilities go beyond just the functionality of calling or sending messages. Thanks to the possibility provided by these devices to connect anywhere, and anytime, users are able to instantaneously share personal and non personal data with other entities in the network. The aggregation and processing of these data allow to describe large-scale phenomena that could not be analyzed with traditional methods, thus paving the way to application scenarios never seen before. This vision is the base of the Social Sensing paradigm, in which the underlying idea is to consider humans as authentic sensors that produce data while interacting with the surrounding environment.

The goal in this research field is to study how human data can be gathered and used to gain situational awareness in a number of socially relevant domains. In this sense, one of the most investigated topic is distributed Human Activity Recognition (HAR). To catch the importance of exploring this research area, it is sufficient to imagine all the services that could benefit from the recognition of human activities, whether it refers to an individual or a group of individuals. Crowd management in emergency situations, improvement of services within smart environments, and health monitoring are just a few examples highlighting its relevance.

Unfortunately, many challenges are still open making difficult, but not impossible, the vision of a world that can fully exploit the potential offered by a distributed HAR system in the real life applications.

This thesis represents an attempt at addressing some of the challenges related to this research area. Starting from the idea that a user implicitly generates data by means of the motion sensors embedded in the smart devices, the design and implementation of a secure and distributed framework for recognizing human activities is described. A preliminary study indicates that some devices, e.g. smartphones, are able to run complex machine learning algorithms to infer activities, while other devices do not allow such processing because it could negatively impact resources, such as the battery life.

In order to overcome this issue, the entire framework is based on the fog-computing paradigm, widely accepted as a reasonable alternative to the cloud when dealing with large amounts of data that need to be processed locally and timely. Then, the activity recognition process leverages on devices that operate at three different logic levels, and are responsible for collecting sensory data, performing HAR, and maintaining the activity models within the community. Each of these tasks is subject to errors that may impact the overall performance of the system. Data collection, for instance, is directly controlled by the user through the smart device, by switching on/off the Android application. As a consequence, it frequently happens that initial and final acquisition windows contain noisy data due to the physical interaction between the user and the device. More generally, data within any window could be altered by unintentional movements, leading to the creation of vocabulary words that are not representative of any micro-activity. To deal with this issue, a noise detection algorithm could be introduced to discard "unreliable" windows before the recognition is performed.

With regards to the HAR process, the combined use of KM-SVM and HMMs allows to obtain a compact representation of sequences of any length, and to dynamically change the set of complex activities to be recognized. One limitation of this schema is that sequences not matching one of the trained HMMs will be associated with the unknown class. Thus, the system is not currently able to automatically recognize (i.e., to correctly name) new activities that may naturally emerge from the community. A future work could focus on the analysis of the unknown set of activities to detect frequent patterns that can be used to train new HMMs on-the-fly.

In addition to the aspects strictly related with the HAR, it was investigated another crucial matter of Social Sensing systems, i.e. the security against cyber attackers. If the data are not protected in any way, a malicious entity may not only degrade the performance of a generic system, but may also target sensitive data of the user for malicious purposes, such as selling them to third parties without the user's consent.

For this reason, an exhaustive discussion on the threats and security requirements of the proposed framework is made. The study highlights that introducing security mechanisms at the different logic layers increases the computational overhead, especially for the resource constrained devices. Then, cyber security mechanisms need to be designed while meeting the computational power of the devices involved. To this aim, it was presented SMCP, a secure protocol for mobile crowdsensing based on Elliptic-Curve Cryptography, Extended Triple Diffie-Hellman Key Agreement, and symmetric cryptography that have been proved to be particularly suitable for mobile smart devices.

The performances of SMCP have been firstly evaluated in terms of computation time required for completing ECC tasks and preparing the messages that implement the secure protocol. Then, a comparative analysis between SMCP and two state-of-the-art protocols, i.e., SAKA and FIRF, has been presented.

Results showed that SMCP takes less time than its competitors to make fog registration, edge registration, and fog-cloud communications secure. This is mainly because of the adoption of the TLS protocol and the X3DHKA protocol. Moreover, although the size of the messages exchanged in SMCP is greater than those used by SAKA and FIRF, the conducted analysis suggests that this small overhead is necessary to overcome some limitations of SAKA and FIRF.

Even though SMCP has been discussed in the case study of a distributed HAR application, the generality of the designed security mechanisms allow to adopt SMCP in other distributed sensing scenarios. For instance, edge devices operating in a smart environment could capture information about the user's presence in order to support a variety of higher-level services, e.g., crowd counting, estimation of people flowing in urban areas, pollution prevention, and so on.

In this scenario, and many others, the reliability of data shared by users plays a key role, since the sharing of false data, especially from synthetic entities, could compromise the functioning of the system. For example, the proposed HAR framework is vulnerable to a such a behavior because the model's refinement is based on data provided by the users, as well as the users' feedback. Hence, if a malicious user starts to send compromised data and/or to release fake feedback on the recognized activities, the whole HAR process will be negatively affected. In the next future the research could investigate how to incentive people for an active and reliable participation. To face this issue, a trust management module could be included to estimate the user's trustworthiness and discourage malicious behaviors.

Bibliography

K. H. Abdulkareem, M. A. Mohammed, S. S. Gunasekaran, M. N. Al-Mhiqani, A. A. Mutlag, S. A. Mostafa, N. S. Ali, and D. A. Ibrahim. A review of fog computing and machine learning: Concepts, applications, challenges, and open issues. *IEEE Access*, 7:153123–153140, 2019.

A. B. Abkenar, S. W. Loke, A. Zaslavsky, and W. Rahayu. Garsaaas: group activity recognition and situation analysis as a service. *Journal of Internet Services and Applications*, 10(1):5, 2019.

V. Agate, F. Concone, and P. Ferraro. Wip: Smart services for an augmented campus. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 276–278, 2018.

C. C. Aggarwal and T. Abdelzaher. *Social Sensing*, pages 237–297. Springer US, Boston, MA, 2013.

H. A. Al Hamid, S. M. M. Rahman, M. S. Hossain, A. Almogren, and A. Alamri. A security model for preserving the privacy of medical big data in a healthcare cloud using a fog computing facility with pairing-based cryptography. *IEEE Access*, 5:22313–22328, 2017.

M. A. Alqarni, S. H. Chauhdary, M. N. Malik, M. Ehatisham-ul Haq, and M. A. Azam. Identifying smartphone users based on how they interact with their phones. *Human-centric Computing and Information Sciences*, 10(1):1–14, 2020.

R. Alvarez, C. Caballero-Gil, J. Santonja, and A. Zamora. Algorithms for lightweight key exchange. *Sensors*, 17(7):1517, 2017.

A. B. Amor, M. Abid, and A. Meddeb. A secure fog-based communication scheme. In *2017 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pages 146–151. IEEE, 2017.

V. Angelakis, I. Avgouleas, N. Pappas, E. Fitzgerald, and D. Yuan. Allocation of heterogeneous resources of an iot device to flexible services. *IEEE Internet of Things Journal*, 3(5):691–700, 2016.

S. Arlot, A. Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas. Window size impact in human activity recognition. *Sensors*, 14(4):6474–6499, 2014.

L. Bao and S. S. Intille. *Activity Recognition from User-Annotated Acceleration Data*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

D. J. Bernstein. Curve25519: new diffie-hellman speed records. In *International Workshop on Public Key Cryptography*, pages 207–228. Springer, 2006.

A. Bonadio, F. Chiti, R. Fantacci, and V. Vespri. An integrated framework for blockchain inspired fog communications and computing in internet of vehicles. *Journal of Ambient Intelligence and Humanized Computing*, 11(2):755–762, 2020.

F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC 1́2, pages 13–16, New York, NY, USA, 2012. ACM.

J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow. Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security*, pages 157–175. Springer, 2014.

J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. *Center for Embedded Network Sensing*, 2006.

Y. Cao, S. Chen, P. Hou, and D. Brown. Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 2–11, Piscataway, NJ, USA, Aug 2015. IEEE.

G. Cardone, A. Cirri, A. Corradi, L. Foschini, and D. Maio. Msf: An efficient mobile phone sensing framework. *Int. Journal of Distributed Sensor Networks*, 9(3):538937, 2013.

G. Cardone, A. Corradi, L. Foschini, and R. Ianniello. Participact: A large-scale crowdsensing platform. *IEEE Transactions on Emerging Topics in Computing*, 4(1):21–32, 2016.

M. A. P. Chamikara, P. Bertók, D. Liu, S. Camtepe, and I. Khalil. Efficient data perturbation for privacy preserving and accurate data stream mining. *Pervasive and Mobile Computing*, 48:1–19, 2018.

B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. Association for Computing Machinery.

D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 4(1):17–32, 2003.

F. Concone, S. Gaglio, G. Lo Re, and M. Morana. *Smartphone Data Analysis for Human Activity Recognition*, pages 58–71. Springer International Publishing, Cham, 2017.

F. Concone, G. L. Re, and M. Morana. A fog-based application for human activity recognition using personal smart devices. *ACM Trans. Internet Technol.*, 19(2), Mar. 2019.

B. Cvetković, V. Janko, A. E. Romero, Ö. Kafalı, K. Stathis, and M. Luštrek. Activity recognition for diabetic patients using a smartphone. *Journal of medical systems*, 40(12):256, 2016.

A. V. Dastjerdi and R. Buyya. Fog computing: Helping the internet of things realize its potential. *Computer*, 49(8):112–116, 2016.

J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

A. De Paola, P. Ferraro, G. L. Re, M. Morana, and M. Ortolani. A fog-based hybrid intelligent system for energy saving in smart buildings. *Journal of Ambient Intelligence and Humanized Computing*, 11(7):2793–2807, 2020.

K. Duan and S. S. Keerthi. Which is the best multiclass svm method? an empirical study. *Multiple classifier systems*, 3541:278–285, 2005.

S. Gaglio, G. Lo Re, and M. Morana. Human activity recognition process using 3-d posture data. *IEEE Transactions on Human-Machine Systems*, 45(5):586–597, Oct 2015.

R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, November 2011.

M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian. Resource management approaches in fog computing: A comprehensive review. *Journal of Grid Computing*, pages 1–42, 2019.

M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi. An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. *Transactions on Emerging Telecommunications Technologies*, 31(2):e3770, 2020.

Google. *Activity Recognition API.* URL `https://developers.google.com/location-context/activity-recognition`.

Google. Activity recognition api. `https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi/`, Nov. 2016.

I. P. W. Group et al. Standard specifications for public key cryptography. *IEEE P1363/D20 (Draft Version 20)*, 2005.

J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction.* Springer, 2 edition, 2009.

K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe. Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC '13, pages 15–20, New York, NY, USA, 2013. Association for Computing Machinery.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

J. Hu, K. Yang, K. Wang, and K. Zhang. A blockchain-based reward mechanism for mobile crowdsensing. *IEEE Transactions on Computational Social Systems*, 7(1):178–191, 2020.

P. Hu, H. Ning, T. Qiu, H. Song, Y. Wang, and X. Yao. Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things. *IEEE Internet of Things Journal*, 4(5):1143–1155, 2017.

M. H. Ibrahim. Octopus: An edge-fog mutual authentication scheme. *IJ Network Security*, 18(6):1089–1101, 2016.

S. Ickin, K. Wac, M. Fiedler, L. Janowski, J. H. Hong, and A. K. Dey. Factors influencing quality of experience of commonly used mobile applications. *IEEE Communications Magazine*, 50(4):48–56, April 2012.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

P. P. Jayaraman, J. B. Gomes, H. L. Nguyen, Z. S. Abdallah, S. Krishnaswamy, and A. Zaslavsky. *CARDAP: A Scalable Energy-Efficient Context Aware Distributed Mobile Data Analytics Platform for the Fog*, pages 192–206. Springer International Publishing, Cham, 2014.

S. Khan, S. Parkinson, and Y. Qin. Fog computing security: a review of current applications and security solutions. *Journal of Cloud Computing*, 6(1):19, 2017.

E. Kim, S. Helal, and D. Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1):48–53, Jan 2010.

S. Kodituwakku and U. Amarasinghe. Comparison of lossless data compression algorithms for text data. *Indian journal of computer science and engineering*, 1 (4):416–425, 2010.

D. I. Kosmopoulos, N. D. Doulamis, and A. S. Voulodimos. Bayesian filter based behavior recognition in workflows allowing for user feedback. *Computer Vision and Image Understanding*, 116(3):422 – 434, 2012. Special issue on Semantic Understanding of Human Behaviors in Image Sequences.

Y. Kwon, K. Kang, and C. Bae. Unsupervised learning for human activity recognition using smartphone sensors. *Expert Systems with Applications*, 41(14):6067–6074, 2014.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials*, 15(3):1192–1209, 2013.

A. Layer. Computer networking: A top down approach. 2017.

J. Lester, T. Choudhury, and G. Borriello. *A Practical Approach to Recognizing Physical Activities*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

J. Li, X. Li, J. Yuan, R. Zhang, and B. Fang. Fog computing-assisted trustworthy forwarding scheme in mobile internet of things. *IEEE Internet of Things Journal*, 6(2):2778–2796, 2018a.

S. Li, Q. Ni, Y. Sun, G. Min, and S. Al-Rubaye. Energy-efficient resource allocation for industrial cyber-physical iot systems in 5g era. *IEEE Transactions on Industrial Informatics*, 14(6):2618–2628, 2018b.

Z. Li, Z. Wei, Y. Yue, H. Wang, W. Jia, L. E. Burke, T. Baranowski, and M. Sun. An adaptive hidden markov model for activity recognition based on a wearable multi-sensor device. *Journal of medical systems*, 39(5):57, 2015.

R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani. A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot. *IEEE Access*, 5:3302–3312, 2017.

L. Lyu, X. He, Y. W. Law, and M. Palaniswami. Privacy-preserving collaborative deep learning with application to human activity recognition. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1219–1228, 2017.

A. Mannini, M. Rosenberger, W. L. Haskell, A. M. Sabatini, and S. S. Intille. Activity recognition in youth using single accelerometer placed at wrist or ankle. *Medicine and science in sports and exercise*, 49(4):801–812, 2017.

P. M. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, USA, 2011.

C. Moore, M. O'Neill, E. O'Sullivan, Y. Doröz, and B. Sunar. Practical homomorphic encryption: A survey. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2792–2795. IEEE, 2014.

A. Munther, R. Razif, M. AbuAlhaj, M. Anbar, and S. Nizam. A preliminary performance evaluation of k-means, knn and em unsupervised machine learning methods for network flow classification. *Journal of Electrical and Computer Engineering*, 6(2):778–784, 2016.

A. A. Mutlag, M. K. Abd Ghani, N. a. Arunkumar, M. A. Mohammed, and O. Mohd. Enabling technologies for fog computing in healthcare iot systems. *Future Generation Computer Systems*, 90:62–78, 2019.

N. Nist. Recommended elliptic curves for federal government use, 1999.

N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta. Comparison of json and xml data interchange formats: a case study. *Caine*, 2009:157–162, 2009.

A. S. Ogale, A. Karapurkar, and Y. Aloimonos. *View-Invariant Modeling and Recognition of Human Actions Using Grammars*, pages 115–126. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

R. Oma, S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa. An energy-efficient model for fog computing in the internet of things (iot). *Internet of Things*, 1:14–26, 2018.

B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik. Fog/edge computing-based iot (feciot): Architecture, applications, and research issues. *IEEE Internet of Things Journal*, 6(3):4118–4149, 2019.

S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers. A review of wearable sensors and systems with application in rehabilitation. *Journal of neuroengineering and rehabilitation*, 9(1):21, 2012.

C. Perera, D. S. Talagala, C. H. Liu, and J. C. Estrella. Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in iot clouds. *IEEE Transactions on Computational Social Systems*, 2(4):171–181, 2015.

C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos. Fog computing for sustainable smart cities: A survey. *ACM Comput. Surv.*, 50(3), June 2017.

D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Machine Learning Technologies*, 2:37–63, 2011.

A. Pratap, F. Concone, V. S. S. Nadendla, and S. K. Das. Three-dimensional matching based resource provisioning for the design of low-latency heterogeneous iot networks. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '19, pages 79–86, New York, NY, USA, 2019. Association for Computing Machinery.

J. R. Quinlan. *C4. 5: programs for machine learning.* Elsevier, 2014.

L. Rabiner and B. Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3*, IAAI'05, pages 1541–1546, Palo Alto, CA, USA, 2005. AAAI Press.

J. D. Rodriguez, A. Perez, and J. A. Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):569–575, 2010.

R. Roman, J. Lopez, and M. Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.

M. S. Ryoo and J. K. Aggarwal. Semantic representation and recognition of continued and recursive human activities. *International journal of computer vision*, 82(1):1–24, 2009.

B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2001.

M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

V. Vo, J. Luo, and B. Vo. Time series trend analysis based on k-means and support vector machine. *Computing and Informatics*, 35(1):111–127, 2016.

J. Vora, S. Tanwar, S. Tyagi, N. Kumar, and J. J. P. C. Rodrigues. Faal: Fog computing-based patient monitoring system for ambient assisted living. In *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6, 2017.

H. Wang, Z. Wang, and J. Domingo-Ferrer. Anonymous and secure aggregation scheme in fog-based public cloud computing. *Future Generation Computer Systems*, 78:712–719, 2018.

M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos. Design of secure key management and user authentication scheme for fog computing services. *Future Generation Computer Systems*, 91:475–492, 2019.

F. Xiao, M. Lu, Y. Zhao, S. Menasria, D. Meng, S. Xie, J. Li, and C. Li. An information-aware visualization for privacy-preserving accelerometer data sharing. *Human-centric Computing and Information Sciences*, 8(1):13, 2018.

G. Xu, H. Li, S. Liu, M. Wen, and R. Lu. Efficient and privacy-preserving truth discovery in mobile crowd sensing systems. *IEEE Transactions on Vehicular Technology*, 68(4):3854–3865, 2019.

J. Yakubu, H. A. Christopher, H. Chiroma, M. Abdullahi, et al. Security challenges in fog-computing environment: a systematic appraisal of current developments. *Journal of Reliable Intelligent Environments*, 5(4):209–233, 2019.

Y. Yao, Y. Liu, Y. Yu, H. Xu, W. Lv, Z. Li, and X. Chen. K-svm: An effective svm algorithm based on k-means clustering. *JCP*, 8(10):2632–2639, 2013.

S. Yi, Z. Qin, and Q. Li. Security and privacy issues of fog computing: A survey. In K. Xu and H. Zhu, editors, *Wireless Algorithms, Systems, and Applications*, pages 685–695, Cham, 2015. Springer International Publishing.

P. Zhang, M. Zhou, and G. Fortino. Security and trust issues in fog computing: A survey. *Future Generation Computer Systems*, 88:16–27, 2018.

X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks and Applications*, 16(3):270–284, 2011.

Y. Zhang, J. Li, D. Zheng, X. Chen, and H. Li. Towards privacy protection and malicious behavior traceability in smart health. *Personal and Ubiquitous Computing*, 21(5):815–830, 2017.