Concone *et al. Hum. Cent. Comput. Inf. Sci.* (2020) 10:28
https://doi.org/10.1186/s13673-020-00232-y

Human-centric Computing and Information Sciences

RESEARCH Open Access

SMCP: a Secure Mobile Crowdsensing Protocol for fog-based applications

Federico Concone, Giuseppe Lo Re and Marco Morana*

*Correspondence: marco.morana@unipa.it
University of Palermo, Viale delle Scienze, ed. 6, Palermo, Italy

Abstract

The possibility of performing complex data analysis through sets of cooperating personal smart devices has recently encouraged 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. Unfortunately, because both of their distributed nature and high degree of modularity, edge-fog-cloud computing systems are particularly prone to cyber security attacks that can be performed against every element of the infrastructure. In order to address this issue, in this paper we present SMCP, a Secure Mobile Crowdsensing Protocol for fog-based applications that exploit lightweight encryption techniques that are particularly suited for low-power mobile edge devices. In order to assess the performance of the proposed security mechanisms, we consider as case study a distributed human activity recognition scenario in which machine learning algorithms are performed by users' personal smart devices at the edge and fog layers. The functionalities provided by SMCP have been directly compared with two state-of-the-art security protocols. Results show that our approach allows to achieve a higher degree of security while maintaining a low computational cost.

Keywords: Human activity recognition, Mobile crowdsensing, Cyber security, Artificial intelligence

Introduction

In recent years, the availability of an ever increasing number of heterogeneous smart devices has changed everyday life as only few other technologies has been able to do in the past.

The diffusion of the Internet of Things (IoT), as well as the spreading of user's personal devices, such as smartphones, tablets, and, more recently, smartwatches or other wristband devices, have enabled new people-centric sensing paradigms in which raw data captured by on board sensors can be analyzed to infer higher-level knowledge. Participatory sensing [1] and mobile crowdsensing [2], for instance, are two common strategies in which the devices owned by a community allow to understand, or even predict, relevant phenomena [3].

In this context, 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 became 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 [4].

In this paper we address the scenario of distributed crowdsensing, and we focus on the design of a secure 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 to continuously transfer 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.

In such a scenario, given the central role of people in the sensing process, preserving sensitive information is mandatory. 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. Moreover, given the low computation capability and the high power consumption of mobile devices, the overhead of making the system secure must be very low.

According to these requirements, we present SMCP, 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.

As compared to existing works, SMCP 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.

In order to evaluate the effectiveness of the proposed protocol we considered as case study a fog-based Human Activity Recognition (HAR) framework that exploits machine

learning algorithms to detect and classify complex activities performed by users wearing a personal smart device. The functionalities provided by SMCP have been firstly compared with other state-of-the-art schemes; then, experimental analysis has focused on direct comparison with two similar security protocols. Results show that SMCP allows to achieve a higher degree of security while maintaining a low computational cost.

The main contributions of this work are as follows. The first is the definition of a novel, lightweight, secure message exchange protocol (SMCP) that covers all the tasks commonly required by a general fog-based crowdsensing application. The second contribution is the adoption and performance evaluation of such a protocol on a real case study in which edge/fog devices are exploited to perform human activity recognition. The third contribution is 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.

The remainder of the paper is organized as follows: related works are outlined in section "Related work". The characteristics of a fog-based architecture for mobile crowdsensing, and the related security requirements are discussed in section "Fog-based architecture for mobile crowdsensing" and section "Requirements and tools for lightweight security", respectively. Section "SMCP: a Secure Mobile Crowdsensing Protocol" describes the preliminary secure enrollment phases provided by SMCP, while its full application in the context of a distributed HAR system is presented in section "Sample Case: secure message exchange for HAR". Experimental results are analyzed in section "Experimental analysis": section "SMCP: performance analysis" provides an assessment of SMCP in terms of message preparation time and size, while a comparative analysis with two other security protocols is discussed in section "SMCP: comparison with SAKA and FIRF protocols". Conclusions follow in section "Conclusion".

Related work

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 [5] 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 [6]. In [7], 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. CARDAP [8], for instance, is a data analytics platform aimed at supporting

mobile crowd-sensing applications in a smart city. The system proposed in [9] 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 [10], 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 [11], 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 [12]. 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 [13], as well as modeling network traffic. An exhaustive analysis of machine learning in the context of fog computing is proposed in [14].

The high degree of modularity of fog systems make them also prone to cyber security attacks that can be performed against every element of the infrastructure [15, 16]. 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 [17]. 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 [18], as well as by implementing lightweight schemes [19] 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 [20], 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 [21] 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 [22] 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 [20, 21] 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 [23], 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 [24] through *permissionless* blockchains in order to guarantee reliability of information collected from individual vehicles. Similarly, blockchain technology is adopted in [25] 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* [26], *data perturbation* [27] and *privacy-preserving data aggregation* [28] techniques can be adopted.

For instance, the privacy leakage problem associated with accelerometer data sharing are highlighted in [29] through an information-aware visualization tool. The authors of [30] 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 [31]. 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 [32]. In [19] 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 [33].

Fog-based architecture for mobile crowdsensing

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, *participatory* sensing, *social* sensing, and *crowd*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 and machine learning, Artificial Intelligence (AI) techniques for 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 idea behind 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 these concepts, this section describes the main features of a human-centric fog-based architecture [34] that can be adopted as basis for a wide number of mobile

crowdsensing applications. Discussion about the architecture and the proposed secure message exchange protocol requires the use of various notations and abbreviations that, for the sake of clarity, are listed in Table 1.

The architecture is composed of three logic levels in which operate devices with increasing computing power. At the lowest level, see Fig. 1, 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

Table 1 List of abbreviations

Abbreviation	Meaning
CDC	Cloud Data Center
ECC	Elliptic Curve Cryptography
ED_i	the *i*-th edge device
FD_i	the *i*-th fog device
FIRF	Face Identification and Resolution Framework
HAR	Human Activity Recognition
KMS	Key Management Server
RSA	Rivest-Shamir-Adleman
SAKA	Secure key management and user authentication scheme
SMCP	Secure Mobile Crowdsensing Protocol
X3DHKA	Extended Triple Diffie–Hellman Key Agreement
Notation	**Meaning**
q	An integer specifying the finite field $\mathbb{F}_q$
FR	The basis used for representing the field elements
$\{a, b\}$	Real numbers $\in \mathbb{F}_q$ defining the equation of the elliptic curve
G	A distinguished point of order n in an elliptic curve group
h	The cofactor of elliptic curve
D	The set of elliptic curve domain parameters
$DH(\cdot)$	Elliptic Curve Diffie–Hellman
(EPU_i, EPR_i)	An ephemeral (public, private) key pair used by entity *i*
(PU_i, PR_i)	A long-term (public, private) key pair used by entity *i*
OPK_i	A set of one-time prekey used by *i*
SPK_i	A momentary rekey signed by entity *i*
$KDF(\cdot)$	A key derivation function
K_S	The established session key
Par_F	Set of parameters describing the characteristics of the fog device
$Cert_F$	A Public-Key-Infrastructure certificate
$Info_F$	A certificate containing the set Par_F
FR_i	A message sent by entity *i* during the fog enrollment phase
ER_i	A message sent by entity *i* during the edge enrollment phase
K_{EC}	A symmetric key for early Edge-Cloud communications
Ω	The set of relevant micro-activities
f_i	The *i*-th feature vector extracted by smart device's embedded sensors
C_i	The *i*-th cluster generated by K-Means
E_{Ei}	The *i*-th message sent by edge device during the secure message exchange phase
E_{Fi}	The *i*-th message sent by fog device during the secure message exchange phase
N_i	The *i*-th nonce

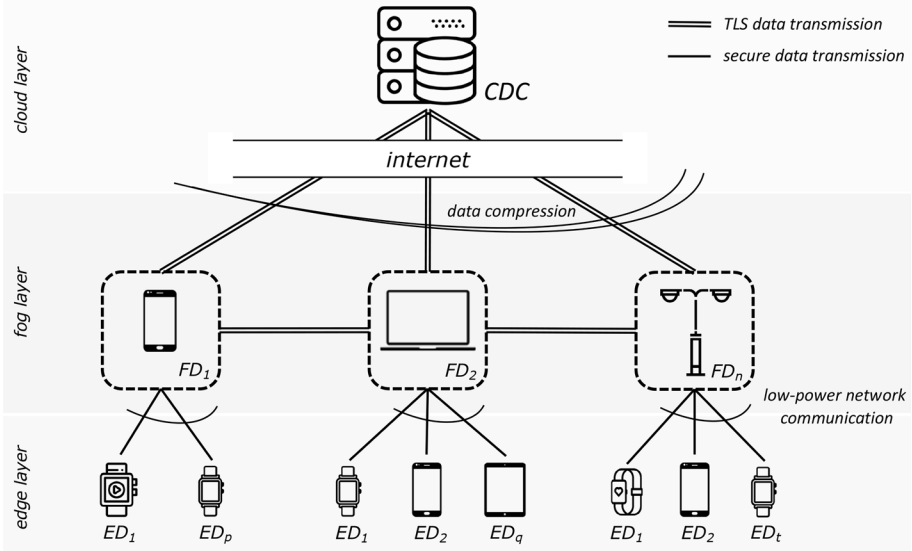

Fig. 1 Fog-based architecture

data to the fog layer. 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 Fig. 1) 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.

Fog devices can also exchange information, e.g., intermediate data models, with other devices at the same layer by means of wireless technologies, such as WiFi, or GSM/xG cellular networks.

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.

Since data processed within the architecture are intended to be kept secret, a lightweight security protocol complements our architecture by guaranteeing secure data transmission and storage.

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.

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 [35]. This feature is crucial to reduce the encryption time, especially when dealing with huge amount of data. In [36], 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 [37, 38], and many standards (such as NIST FIPS 186-2 [39] and IEEE P1363 [40]) 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 [41]. 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:

- PU_A, PU_B: long-term public keys of the two entities;
- EPU_A, EPR_A: an ephemeral key pair used by A;
- SPK_B: a momentary prekey signed by B that will be updated at some interval (e.g. once a week, or once a month);
- 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 PU_B, the signed prekey SPK_B, the digital signature on SPK_B, and a set of m one-time prekeys $S = \{OPK_{B1}, OPK_{B2}, \ldots, OPK_{Bm}\}$.

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 (EPU_A, EPR_A). The 32 bytes session key K_S is obtained by calculating the following functions:

$$KDF(DH(PU_A, SPK_B) \,||\, DH(EPU_A, PU_B) \,||\, DH(EPU_A, SPK_B) \,||\, DH(EPU_A, OPK_B)),$$

where KDF is the HMAC-based key derivation function described in [42], and the generic $DH(PU_1, PU_2)$ 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 PU_A, EPU_A, the value i that identifies the one-time prekey OPK_{Bi} used by A, and an initial message encrypted with the session key K_S. The entity B will use such information to calculate the DH and KDF functions and derive the key 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.

SMCP: a Secure Mobile Crowdsensing Protocol

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 enrollment* 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 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 [43]. The next subsections describe the entities, the messages exchanged, and the actions needed to implement the SMCP protocol.

Fog enrollment

A new fog device can be added to the system by installing on it the *fogApp* (Fig. 2—step 1). In order to support secure communications, this app includes the elliptic

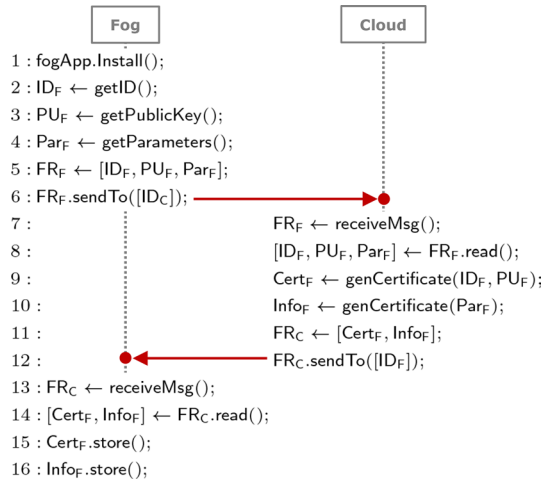

Fig. 2 Fog devices enrollment

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

The computational capabilities of fog and cloud devices make them able to implement the TLS protocol; thus, both FR_F and 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 $\{PU_F, PR_F\}$, and the pair of certificates $\{Cert_F, Info_F\}$ (steps 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 "Efficient key exchange on mobile smart devices".

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* (Fig. 3— step 1) includes the fog discovery routines, a symmetric key K_{EC} for early Edge-Cloud communications, and the cloud public key PU_C.

In an early phase, the fog announces its presence by broadcasting a message containing the fog ID_F and the pair of certificates obtained during the fog registration step (step 4). Such information is used by the edge to verify the identity of the fog (step 6), and to send the request for downloading the full version of the *edgeApp* (step 11).

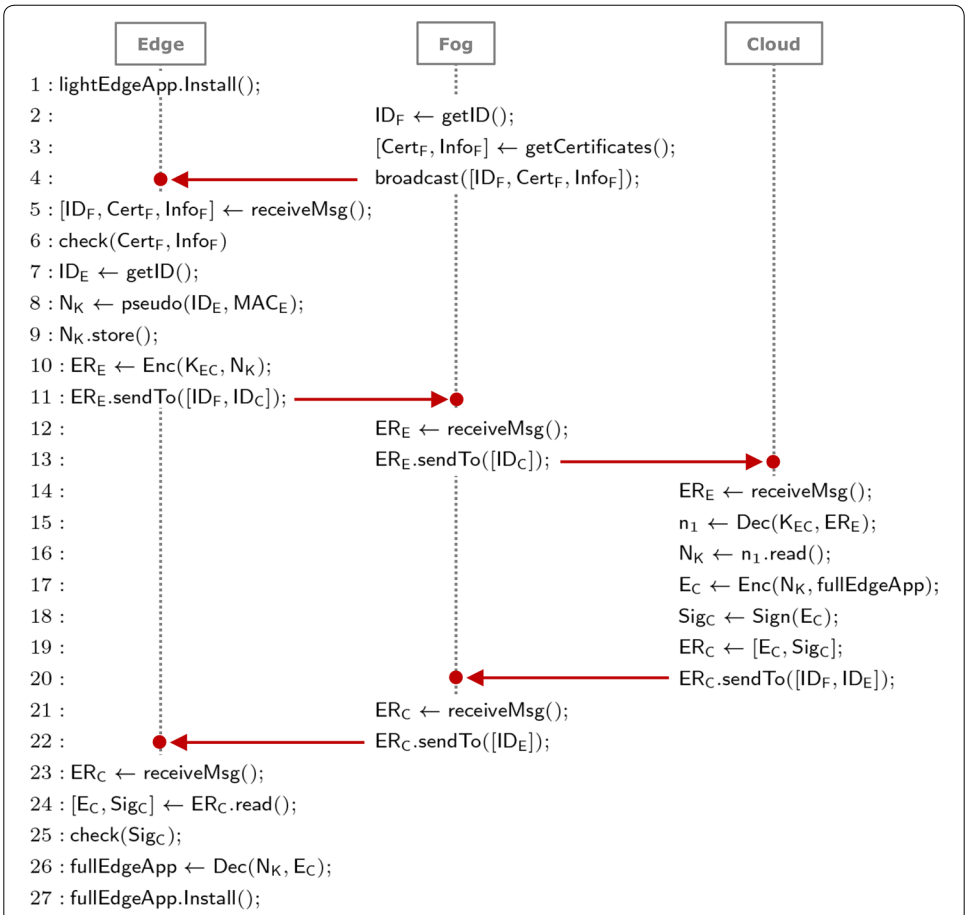

Fig. 3 Edge devices enrollment

To this aim, the edge sends to the CDC (by relying on the fog $\mathsf{ID_F}$) the message $\mathsf{ER_E}$ (step 13) containing a pseudorandom number computed as function of the edge identifier $\mathsf{ID_E}$ and the physical address of the device.

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

In addition to the fog and edge enrollment, SMCP covers all the phases in which the crowdsensing algorithms are executed actually. To prove the validity of the proposed approach, we considered as case study the definition of a secure fog-based HAR framework aiming to perform near real-time recognition of activities of different lengths.

Sample case: secure message exchange for HAR

In this section, we first provide a brief description of the algorithms that implement the HAR system; then, we present how SMCP is used to protect data exchanged during edge-fog, and fog-cloud communications needed to perform the HAR process.

Overview of the HAR framework

The goal of the HAR framework we adopted here [34] is to perform near real-time recognition of *complex* activities of different lengths. The main idea is that *complex* activities can be reasonably decomposed in *simple*, atomic, micro-activities. For instance, for a given user, the everyday activity *go to work* may be performed by (1) *walking* for a while, (2) driving or being in a *vehicle* for a certain amount of time, then (3) *walking* again, (4) going up the *stairs*, and finally arriving at the office (5) staying *still*. The most representative characteristics of these five phases can be easily captured by the sensors embedded in the smartwatches; on the contrary, an overall knowledge of the activity carried out is more difficult to obtain. Thus, we can start by building up feature vectors that summarize data from the three-axis accelerometer and gyroscope sensors, i.e., by computing their *max, min, mean, standard deviation, root mean square* values at fixed time intervals [44]. In particular, considering both acceleration and angular velocity values allows to discriminate between micro-activities that exhibit similar acceleration values (e.g., *sitting* or *driving*), although being characterized by different types of oscillations.

Given that each feature vector captures the hidden characteristics of a certain micro-activity *ma*, a complex activity *CA* can be seen as a specific sequence of micro-activities $\{ma_1, ma_2, \ldots, ma_n\}$. In order to find a minimal set of Ω relevant micro-activities, with $\Omega << n$, which can properly describe every *CA* we want to recognize, two machine learning algorithms, namely, K-means clustering and SVM classifiers, can be combined. This approach, known as KM-SVM or CSVM, exploits K-Means to group the feature vectors $(f_1, f_2, \ldots, f_n)$ into a lower number of clusters $C = (C_1, C_2, \ldots, C_\Omega)$, that will be used to train the SVMs.

After CSVM has been performed, the resulting set $\{ma_1, ma_2, \ldots, ma_\Omega\}$ can be exploited to model any complex activity. To this aim, this collection is used to train m distinct Hidden Markov Models, where m is the number of complex activities the system can recognize. In our scenario, the activities can be seen as the hidden states we want to find according to a set of observations, i.e., sensor data. Thus, given a set of micro-activities, the recognition of a new, unknown, sequence is performed by testing it against all the HMMs, and selecting the class associated with the largest posterior probability.

SMCP for HAR

In order to protect data exchanged during the HAR phases, edge and fog devices must agree on a secret session key $\mathsf{K_S}$ using the X3DHKA protocol.

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 $\mathsf{EPU_E}$ (Fig. 4a, step 6). Ciphering $\mathsf{EPU_E}$ with the public key of the fog prevents an intruder to know the seed from which it is possible to generate $\mathsf{K_S}$, while the digital signature $\mathsf{Sig_{EPU}}$ guarantees that $\mathsf{EPU_E}$ is authentic. After receiving $\mathsf{E_{E1}}$, the fog checks $\mathsf{Sig_{EPU}}$ and computes the session key $\mathsf{K_S}$ using the X3DHKA scheme. Now only the fog device knows $\mathsf{K_S}$; thus, in order to let the edge device to compute the session key, the message $\mathsf{E_{F1}}$ is sent (step 15). This message specifies both the one-time prekey $\mathsf{OPK_F}$ 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.

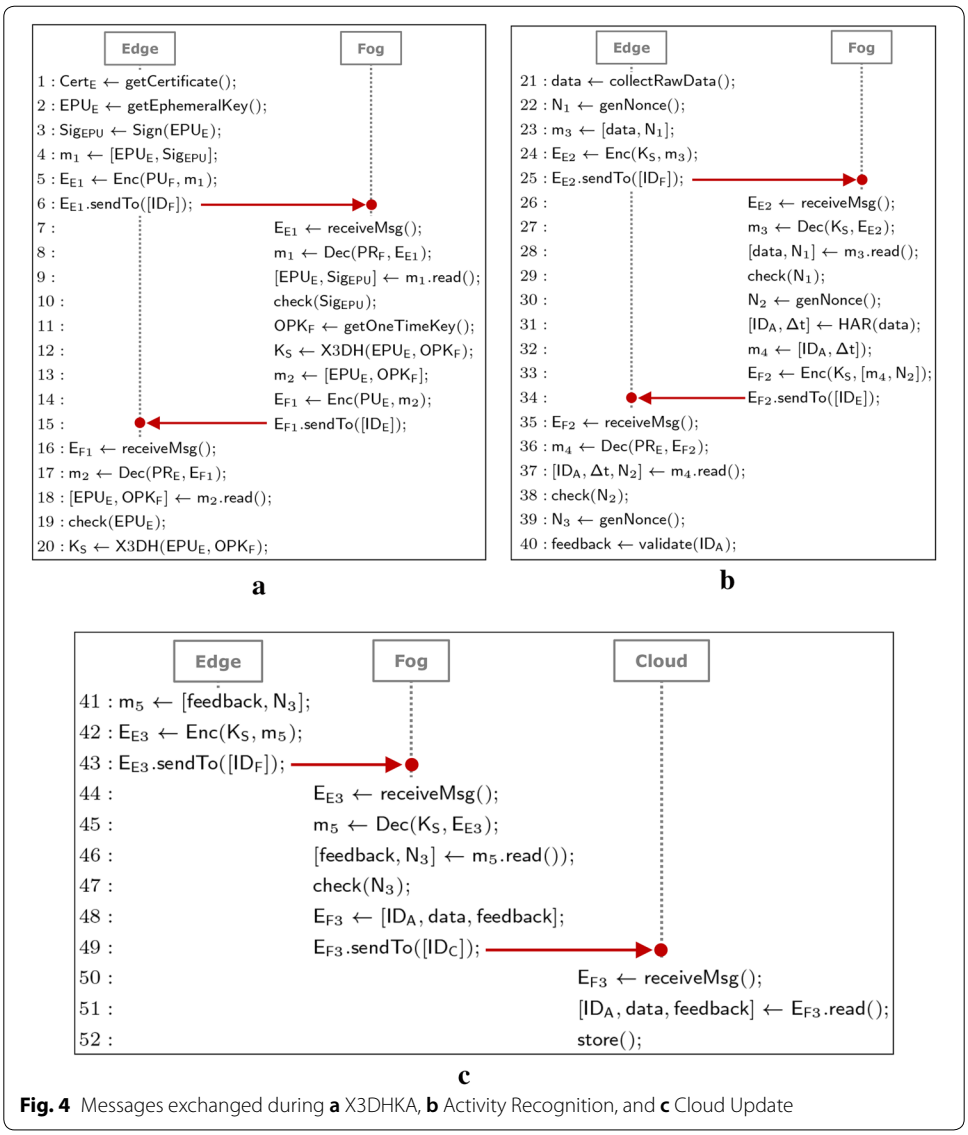

Fig. 4 Messages exchanged during **a** X3DHKA, **b** Activity Recognition, and **c** Cloud Update

When the edge device receives E_{F1}, it verifies that EPU_E corresponds to the one reported in message E_{E1} and performs all the steps needed to obtain K_S (steps 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 (Fig. 4b, step 25). The field data includes both the feature vector computed on the accelerometer and gyroscope measures, and the timestamp reporting when data were captured. Moreover, in order to face a replay attack within the established session between edge and fog, E_{E2} also contains a nonce N_1.

The messages of type 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 E_{F2} containing the ID_A of the activity, the time interval within it

Table 2 Devices used in the proposed case study

	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	1.0	1.2	1.8	2.0	3.9	*cloud*
# core	4	2	4	8	8	4	*computing*
RAM (MB)	512	768	512	3072	4096	8192	*web service*

was performed, and a nonce N_2 ensuring the uniqueness of the message (step 34). To improve the performance of the HAR system, the end user can provide a *feedback* about the output of the recognition process (see Fig. 4c). To this aim, a message E_{E3} is sent from edge device to fog device (step 43). Finally, the message E_{F3} including all data involved in the activity recognition process is transferred from the fog to the cloud through a secure TLS channel (step 49), and then stored (step 52).

It is worth noting that X3DH Key Agreement (Fig. 4a) is common to every kind of application in which SMCP is adopted. On the contrary, different crowdsensing applications may require the protection (Fig. 4b) and synchronization (Fig. 4c) 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 m_3 should be modified to contain other sensory data, such as accelerometer or GPS information. Then, many m_3 messages would be sent to the fog where an ad-hoc machine learning algorithm (step 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 (step 40).

Experimental analysis

In order to assess the validity of the proposed approach, we 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.

All the experiments described below were conducted by using as edge nodes various Android-based smartphones and smartwatches equipped with gyroscope and accelerometer sensors. Data between smartwatches and smartphones are exchanged through short-range Bluetooth technology, but the proposed architecture is compatible with a number of smart devices which provide other wireless communication interfaces.

In order to support the processing steps described so far, each mobile device must came at least with a dual-core processor and 512 MB RAM. Two mobile applications for smartphones and smartwatches were developed. The smartphone app is suited for Android 4.0 *Ice Cream Sandwich* OS or higher, whilst the smartwatches require at least Android 4.1 *Jelly Bean* OS.

In order to validate the generality of the architecture, we considered a test bed consisting of several devices, each of them having distinct hardware capabilities (see Table 2). In particular, we used (i) three different models of smartwatches as edge devices, (ii) two

types of smartphones and one laptop as fog devices, and (iii) a commercial cloud computing web server as CDC.

The assessment of the HAR process described in section "Overview of the HAR framework" was made asking 20 volunteers to perform 10 different complex activities, such as *go-to-work* (walking for a while), *jogging* (alternating running and walking phases), *cooking* (briefly moving in the kitchen), and *driving* (staying in a vehicle for a long time). Preliminary experiments were intended to find the best set of parameters for the activity recognition procedure, that is the number of clusters/words in the dictionary (Ω), and the number of hidden states in HMMs (N). To this aim, grid-search and 10-fold cross validation on the pair (Ω, N) were performed. Once the best configuration for the HAR system was established, other experiments were conducted to evaluate the recognition performance on a new test set. A comprehensive evaluation of the HAR framework is reported in [34].

SMCP: performance analysis

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 5 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

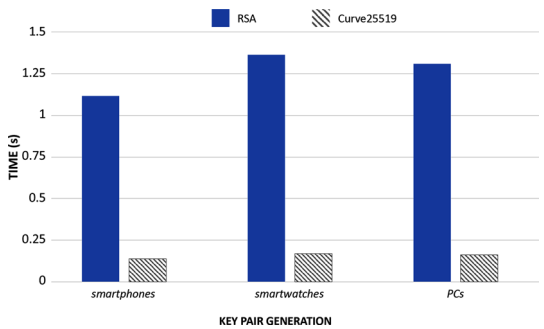

Fig. 5 Average computation time required to generate the key pair by RSA and Curve25519 on smartphones, smartwatches, and PCs

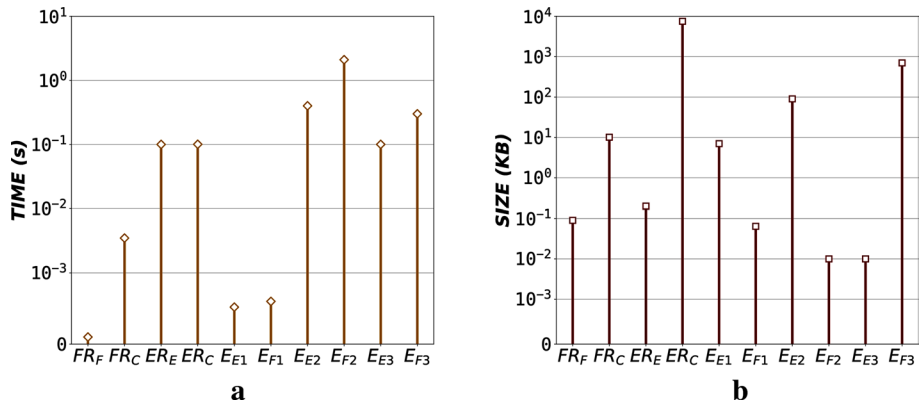

Fig. 6 Message complexity in terms of computation time (**a**) and size (**b**)

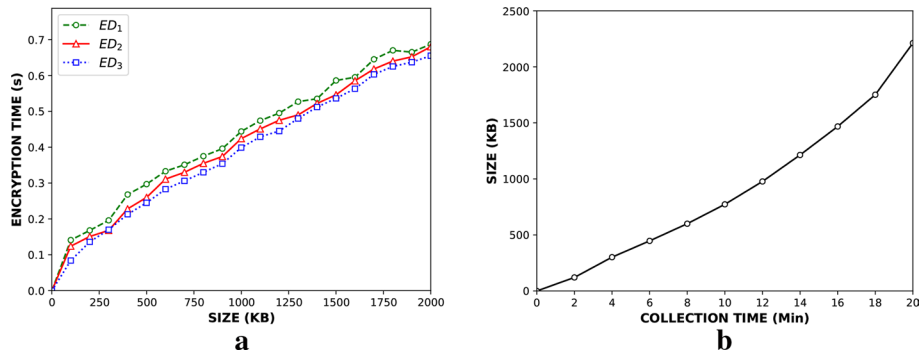

Fig. 7 **a** Size of E_{E2} while varying the data collection time, and **b** the corresponding average encryption time using AES

complexity of the messages described in section "SMCP for HAR" in terms of computation time required for their preparation t_p, and size s.

Figure 6a 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 $t_p(E_{F2})$ 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, we can 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 Fig. 6b. 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 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 2. The curves depicted in Fig. 7a show that the time required for encrypting the message E_{E2} is similar for the three edge devices we adopted. This result is easily explainable since AES is efficiently performed in hardware in almost any recent smart device. According to these results, we chose to limit the size of E_{E2} to about 100 KB so as to perform AES encryption in about 100 ms. Moreover, as shown in Fig. 7b, such an amount of data is collected in about 2 min when using the same sampling frequency for all the devices.

SMCP: comparison with SAKA and FIRF protocols

In order to present a preliminary comparison of SMCP with other approaches representing the state-of-the-art, Table 3 summarizes the security features provided by some relevant related protocols. We can notice that some of them, i.e., [20] and [21], 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 we present a comparative analysis of SMCP with a general-purpose protocol, namely SAKA [19], and a specific application domain protocol, FIRF [30].

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 we 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, we followed the same approach adopted by the authors of SAKA. We started from the sequence of messages used in SMCP (as described in section "SMCP for HAR"), and

Table 3 Security features comparison of SMCP with some other protocols

Feature	SMCP	[19]	[20]	[21]	[30]	[31]
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	✓	✓			✓	

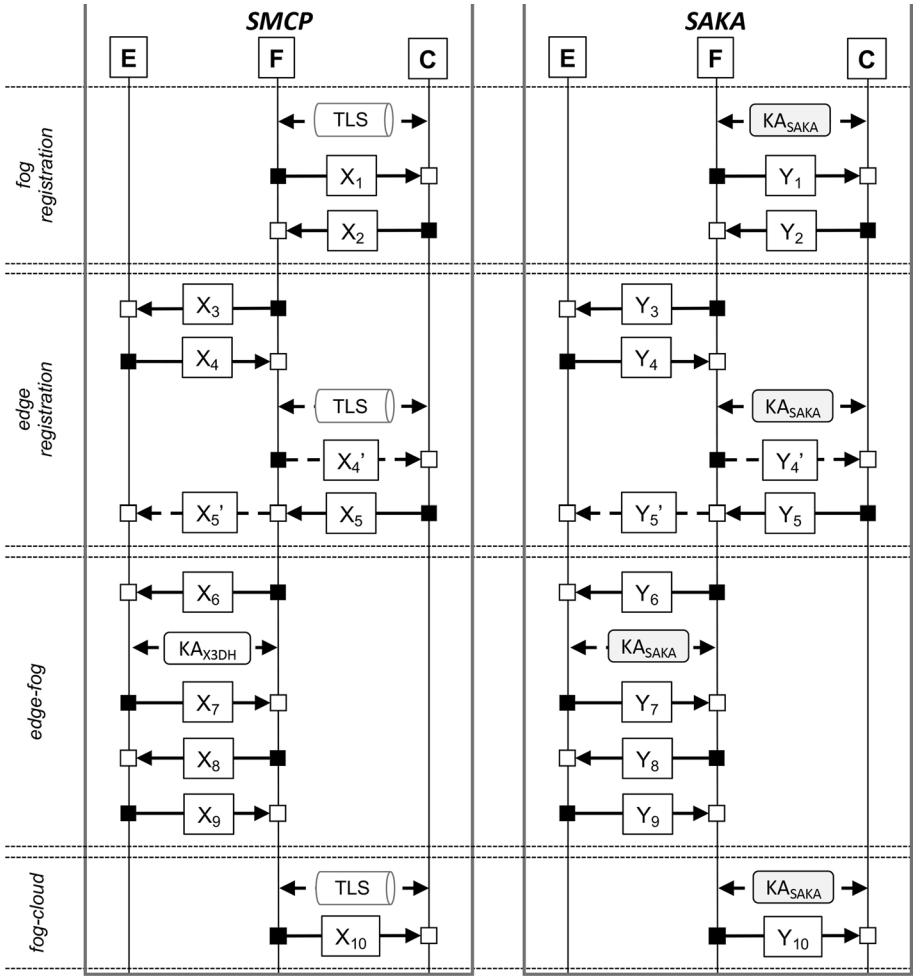

Fig. 8 Messages exchanged during four phases common to the SMCP and SAKA protocols

Table 4 Comparison of SMCP and SAKA computation costs

	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	$3T_{tls} + 3T_{cert} + 9T_{sed} + T_{sig} + T_{x3dh}$	$3T_{kafc} + 9T_{sed} + T_{sig} + T_{kaef}$

we 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 Fig. 8) that provide the same functionalities of SMCP while using the SAKA key agreement scheme. Then, we 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 generation), 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, highlight that some computations are common to both protocols, i.e., symmetric encryption and decryption (sed) and signature generation (sig), while others reflect 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 our 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 ($Cert_F$ and $Cert_E$) 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.

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 [30]. 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 5.

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

Table 5 SMCP and FIRF compared in terms of average computation time (s) and size (KB) of the messages exchanged during the fog-cloud communication

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

covered by FIRF due to the lack of secure communication between edge and fog devices, as discussed in section "Related work".

Conclusion

In this paper we focused on the threats and security requirements of a fog-based crowdsensing framework in which the processing tasks are distributed among different types of users' smart devices. Due to the intrinsic distributed nature of the fog environment, the system is prone to attacks intended to obtain information about the users, or alter the performance of the system.

For this reason, cyber security mechanisms need to be designed while meeting the computational power of the devices involved. To this aim, we 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 the 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 analysis we have conducted suggests that this small overhead is necessary to overcome some limitations of SAKA and FIRF.

Even though our system has been discussed in the case study of a distributed HAR application, the generality of the security mechanisms we designed allow to 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.

As regards future work, being the whole sensing process based on the user's reliability, one of the main limitations of the proposed security scheme is the protection from internal attacks performed by legitimate users. This point is crucial to any

crowdsensing system and can be faced by designing incentives to reliable participation [45]. To this aim, we will focus on reputation management mechanisms that would allow to estimate the user's trustworthy and discourage malicious behaviors.

Acknowledgements
Not applicable.

Authors' contributions
All authors provided equal contribution to the study. FC focused on the definition of the fog-based architecture and performed the experimental analysis. GLR supervised the whole study and conducted the analysis of the security requirements. MM designed the secure protocol, defined the experimental campaign and interpreted its results. All authors read and approved the final manuscript.

Funding
This research is partially funded by the Crowdsense project - PO FESR 2014-2020 cup G29J18000740007.

Availability of data and materials
The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request.

Competing interests
The authors declare that they have no competing interests.

Received: 1 February 2020 Accepted: 29 April 2020
Published online: 02 July 2020

References

1. Lane ND, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell AT (2010) A survey of mobile phone sensing. IEEE Commun Mag 48(9):140
2. Ganti RK, Ye F, Lei H (2011) Mobile crowdsensing: current state and future challenges. IEEE Commun Mag 49(11):32–39
3. Concone F, Ferraro P, Lo Re G (2018) Towards a smart campus through participatory sensing. In: 2018 IEEE International Conference on Smart Computing (SMARTCOMP), pp 393–398
4. Omoniwa B, Hussain R, Javed MA, Bouk SH, Malik SA (2019) Fog/edge computing-based IoT (FECIoT): architecture, applications, and research issues. IEEE Internet Things J 6(3):4118–4149. https://doi.org/10.1109/JIOT.2018.2875544
5. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the Internet of Things. In: Proc. of the First Edition of the MCC Work. on Mobile Cloud Computing. MCC '12, pp 13–16. ACM, USA
6. Ghobaei-Arani M, Souri A, Rahmanian AA (2019) Resource management approaches in fog computing: a comprehensive review. J Grid Comput. 18:1–42
7. Ghobaei-Arani M, Souri A, Safara F, Norouzi M (2020) An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. Trans Emerg Telecommun Technol 31(2):3770. https://doi.org/10.1002/ett.3770.e3770ETT-18-0545.R2
8. Jayaraman PP, Gomes JB, Nguyen HL, Abdallah ZS, Krishnaswamy S, Zaslavsky A (2014) Cardap: a scalable energy-efficient context aware distributed mobile data analytics platform for the fog. Lecture Notes in Comput Sci 8716:192–206
9. De Paola A, Ferraro P, Lo Re G, Morana M, Ortolani M (2019) A fog-based hybrid intelligent system for energy saving in smart buildings. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-019-01375-2
10. Perera C, Talagala DS, Liu CH, Estrella JC (2015) Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in IoT clouds. IEEE Trans Comput Soc Syst 2(4):171–181
11. Cao Y, Chen S, Hou P, Brown D (2015) Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In: 2015 IEEE Int. Conf. on Networking, Architecture and Storage (NAS), pp 2–11
12. Mutlag AA, Ghani MKA, Arunkumar N, Mohammed MA, Mohd O (2019) Enabling technologies for fog computing in healthcare IoT systems. Future Gener Comput Syst 90:62–78. https://doi.org/10.1016/j.future.2018.07.049
13. Oma R, Nakamura S, Duolikun D, Enokido T, Takizawa M (2018) An energy-efficient model for fog computing in the Internet of Things (IoT). Internet Things 1–2:14–26. https://doi.org/10.1016/j.iot.2018.08.003
14. Abdulkareem KH, Mohammed MA, Gunasekaran SS, Al-Mhiqani MN, Mutlag AA, Mostafa SA, Ali NS, Ibrahim DA (2019) A review of fog computing and machine learning: concepts, applications, challenges, and open issues. IEEE Access 7:153123–153140. https://doi.org/10.1109/ACCESS.2019.2947542
15. Roman R, Lopez J, Mambo M (2018) Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Gener Comput Syst 78:680–698. https://doi.org/10.1016/j.future.2016.11.009
16. Khan S, Parkinson S, Qin Y (2017) Fog computing security: a review of current applications and security solutions. J Cloud Comput 6(1):19. https://doi.org/10.1186/s13677-017-0090-3
17. Zhang P, Zhou M, Fortino G (2018) Security and trust issues in fog computing: a survey. Future Gener Comput Syst 88:16–27. https://doi.org/10.1016/j.future.2018.05.008
18. Alqarni MA, Chauhdary SH, Malik MN, Ehatisham-ul-Haq M, Azam MA (2020) Identifying smartphone users based on how they interact with their phones. Human-centric Comput Inf Sci 10(1):7. https://doi.org/10.1186/s13673-020-0212-7
19. Wazid M, Das AK, Kumar N, Vasilakos AV (2019) Design of secure key management and user authentication scheme for fog computing services. Future Gener Comput Syst 91:475–492. https://doi.org/10.1016/j.future.2018.09.017

20. Ben Amor A, Abid M, Meddeb A (2017) A secure fog-based communication scheme. In: 2017 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC), pp. 146–151. https://doi.org/10.1109/IINTEC.2017.8325929
21. Ibrahim MH (2016) Octopus: an edge-fog mutual authentication scheme. Int J Netw Secur 18(6):1089–1101. https://doi.org/10.6633/IJNS.201611.18(6).10
22. Li J, Li X, Yuan J, Zhang R, Fang B (2019) Fog computing-assisted trustworthy forwarding scheme in mobile Internet of Things. IEEE Internet Things J 6(2):2778–2796. https://doi.org/10.1109/JIOT.2018.2874808
23. Xu G, Li H, Liu S, Wen M, Lu R (2019) Efficient and privacy-preserving truth discovery in mobile crowd sensing systems. IEEE Trans Veh Technol 68(4):3854–3865. https://doi.org/10.1109/TVT.2019.2895834
24. Bonadio A, Chiti F, Fantacci R, Vespri V (2020) An integrated framework for blockchain inspired fog communications and computing in internet of vehicles. J Ambient Intell Humaniz Comput 11(2):755–762. https://doi.org/10.1007/s12652-019-01476-y
25. Hu J, Yang K, Wang K, Zhang K (2020) A blockchain-based reward mechanism for mobile crowdsensing. IEEE Trans Comput Soc Syst 7(1):178–191. https://doi.org/10.1109/TCSS.2019.2956629
26. Zhang Y, Li J, Zheng D, Chen X, Li H (2017) Towards privacy protection and malicious behavior traceability in smart health. Pers Ubiquitous Comput 21(5):815–830
27. Chamikara MAP, Bertok P, Liu D, Camtepe S, Khalil I (2018) Efficient data perturbation for privacy preserving and accurate data stream mining. Pervasive Mobile Comput 48:1–19
28. Lu R, Heung K, Lashkari AH, Ghorbani AA (2017) A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT. IEEE Access 5:3302–3312
29. Xiao F, Lu M, Zhao Y, Menasria S, Meng D, Xie S, Li J, Li C (2018) An information-aware visualization for privacy-preserving accelerometer data sharing. Human-centric Comput Inf Sci. https://doi.org/10.1186/s13673-018-0137-6
30. Hu P, Ning H, Qiu T, Song H, Wang Y, Yao X (2017) Security and privacy preservation scheme of face identification and resolution framework using fog computing in Internet of Things. IEEE Internet Things J 4(5):1143–1155. https://doi.org/10.1109/JIoT.2017.26597
31. Wang H, Wang Z, Domingo-Ferrer J (2018) Anonymous and secure aggregation scheme in fog-based public cloud computing. Future Gener Comput Syst 78:712–719. https://doi.org/10.1016/j.future.2017.02.032
32. Moore C, O'Neill M, O'Sullivan E, Doröz Y, Sunar B (2014) Practical homomorphic encryption: A survey. In: 2014 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2792–2795. https://doi.org/10.1109/ISCAS.2014.6865753
33. Yakubu J, Abdulhamid SM, Christopher HA, Chiroma H, Abdullahi M (2019) Security challenges in fog-computing environment: a systematic appraisal of current developments. J Reliab Intell Environ 5(4):209–233. https://doi.org/10.1007/s40860-019-00081-2
34. Concone F, Lo Re G, Morana M (2019) A fog-based application for human activity recognition using personal smart devices. ACM Trans Internet Technol 19(2):20–12020. https://doi.org/10.1145/3266142
35. Alvarez R, Caballero-Gil C, Santonja J, Zamora A (2017) Algorithms for lightweight key exchange. Sensors 17:7. https://doi.org/10.3390/s17071517
36. Hamid HAA, Rahman SMM, Hossain MS, Almogren A, Alamri A (2017) 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
37. Hankerson D, Menezes A (2011) In: van Tilborg HCA, Jajodia S (eds.) Elliptic Curve Cryptography, pp 397–397. Springer, Boston, MA
38. Bos JW, Halderman JA, Heninger N, Moore J, Naehrig M, Wustrow E (2014) Elliptic curve cryptography in practice. Int Conf on Financial Cryptography and Data Security. Springer, Cham, pp 157–175
39. NIST (1999) Recommended Elliptic Curves for Federal Government Use. http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf
40. Group IPW, et al (1999) Ieee p1363: Standard specifications for public key cryptography. http://grouper.ieee.org/groups/1363
41. Bernstein DJ (2006) Curve25519: new Diffie–Hellman speed records. Int work on public key cryptography. Springer, Berlin, pp 207–228
42. Krawczyk H (2010) Cryptographic extraction and key derivation: The hkdf scheme. In: Rabin T (ed) Advances in cryptology - CRYPTO 2010. Springer, Berlin, Heidelberg, pp 631–648
43. Kurose JF, Ross KW (2012) Computer networking: a top-down approach (6th Edition), 6th edn. Pearson, UK
44. Concone F, Gaglio S, Lo Re G, Morana M (2017) Smartphone data analysis for human activity recognition. AI*IA 2017 advances in artificial intelligence. Springer, Cham, pp 58–71
45. Ogie RI (2016) Adopting incentive mechanisms for large-scale participation in mobile crowdsensing: from literature review to a conceptual framework. Human-centric Comput Inf Sci. https://doi.org/10.1186/s13673-016-0080-3

Publisher's Note