Secure Monitoring in IoT-Based Services via Fog Orchestration

Alexandre Viejo, David Sánchez[1]

Universitat Rovira i Virgili, Department of Computer Science and Mathematics, UNESCO Chair in Data Privacy, CYBERCAT-Center for Cybersecurity Research of Catalonia, Av. Països Catalans 26, 43007 Tarragona, Spain
E-mail: {alexandre.viejo,david.sanchez}@urv.cat

Abstract

Fog computing proposes moving computation, communication and storage from the cloud to the edge of the network, thus making it a perfect match for latency-sensitive IoT services such as sensor monitoring. Security is, however, a fundamental issue in fog-enabled services dealing with sensitive data, such as healthcare monitoring. Some authors circumvent security issues by assuming that IoT devices and fog nodes are fully trusted. Nevertheless, when IoT devices are geographically distributed and/or may be deployed in non-secure locations, and communications are done via WAN, the probability of attacks increases significantly. Security-enabling solutions rely on either trusted and centralized infrastructures (which go against the distributed and ubiquitous nature of fog-enabled IoT services) or expensive cryptography (which increases latency). In contrast, we propose decentralized and efficient security-enabling protocols for fog-based services that involve continuous monitoring of data. We have put special care in ensuring that the performance benefits brought by fog computing are not significantly hampered by the security mechanisms in use. For this, we rely on the recent concept of *fog orchestration* by which the network is self-tailored to the service to be delivered; in this way, we minimize the load of the network and its nodes during the monitoring. By means of a set of theoretical analyses and empirical results, we discuss the performance of our protocols and how they improve the current literature.

Keywords: Security, Internet of things, Fog orchestration, Data monitoring.

1. Introduction

Fog computing has recently emerged as an infrastructure paradigm in which a substantial amount of computation, communication and short term storage

[1]Corresponding author. Address: Department of Computer Science and Mathematics, Universitat Rovira i Virgili, Av. Països Catalans 26, 43007 Tarragona, Spain. Tel.:+34 977 559657; Fax: +34 977 559710. E-mail: david.sanchez@urv.cat

are moved from the cloud service provider to devices located at the edge of the network. This adds several benefits, such as enhanced mobility, low latency and reduced network bandwidth, thus making fog computing a perfect match for latency-sensitive or real-time applications [1]. Paradigmatic examples of such applications can be found in the healthcare domain. Emergency and health monitoring rely on complex networks of interconnected sensor nodes and devices that continuously send large amounts of data to dedicated servers or to the cloud [2]. Because sensor nodes are typically low tier and heterogeneous IoT devices that may be geographically distributed, a standard sensor-to-cloud infrastructure is neither practical nor efficient. Instead, with fog computing, smaller sensor networks are managed by devices located at the edge of the network that act as gateways towards the cloud. These latter devices, which are usually fully fledged ones (with respect to computation and storage), are known as *fog nodes*. Due to the good match between the needs of IoT-based healthcare services and the advantages brought by fog computing, a variety of fog-enabled healthcare services have been recently proposed [1, 2]. The most common ones consider sensor networks and devices located in (potentially distributed) medical facilities [3, 4], but there have been also examples of services meant for mobile and home treatment [5, 6]. In these latter cases, sensors are deployed on subject basis (e.g., via smart watches or smart shirts) and the edge devices correspond to the patient's smartphone or router. A constant of all these applications is that they imply continuous *monitoring* of the data sensed by the IoT devices. Upcoming Industry 4.0 (i.e., Industrial Internet-of-Things) applications [7, 8] are also Fog-enabled and have similar needs: real-time monitoring, ultra-low latency and reliability.

Fog computing is, however, not free from challenges. Security and privacy arise as fundamental issues when dealing with highly sensitive data such as healthcare data [9]. Some authors circumvent security issues by assuming that IoT devices and fog nodes located at the patients' or hospital's premisses are fully trusted, and by considering privacy issues only towards the cloud but not inside the local network [10, 11, 12]. Others employ dedicated trusted hardware or symmetric key cryptography to protect the data, but they only support a limited and simple set of predefined services [13, 14]. Nevertheless, when IoT devices are geographically distributed (as it happens in complex healthcare services) and/or may be deployed in non-secure locations (such as patients' premises), and communications are done via WAN, the probability of attacks increases significantly. This is specially true for IoT devices that, due to their limited hardware, are specially vulnerable to cyber- or even physical attacks. Because these devices may store and transmit sensitive data (e.g., patients' vital signs), serious privacy threats arise when IoT devices cannot be fully trusted.

1.1. Contributions and plan of this paper

This work tackles the issues discussed above by proposing secure and privacy-by-design protocols for fog-based services that involve monitoring of data generated by IoT devices. Because enhanced security and privacy typically imply

overheads (e.g., due to the cryptographic operations involved), we have put special care in ensuring that the performance benefits brought by fog computing (efficiency, low latency and scalability) are not significantly hampered by the implemented security mechanisms. For this, we rely on the recent concept of *fog orchestration* [15], which consists in matching the devices' and network's operation with the service needs. Specifically, fog orchestration considers that most services only require the communication of a subset of devices from the whole network (e.g., monitoring the sensors of a specific patient within a concrete hospital department). Consequently, through network orchestration, it is possible to determine and select the appropriate IoT appliances to dynamically compose a simplified workflow for the requested monitoring service. This allows devices not directly involved in the service to save their resources; on top of that, managing a smaller set of devices reduces the amount of data that is (re-)transmitted through the network. We also employ fog orchestration innovatively to create and distribute the security-enabling cryptographic materials among the involved nodes, which are tailored for efficient monitoring. In this way, whereas the orchestration step, which implies multicasting the service request, is secured by relying on Attribute-Based Encryption (ABE), the data monitoring process, which is run on the orchestrated network, is mainly based on highly efficient symmetric cryptography.

Focusing on the use of costly cryptography such as ABE on the orchestration process, which runs seldom, and using lightweight cryptography in the monitoring process, which runs continuously, allows our approach to minimize the latency and energy consumption of IoT devices during the run of a monitoring service. This differentiates our approach from related works that, due to the lack of orchestration, employ expensive ABE [16, 17] or even homomorphic encryption [18] during the whole life cycle of the service.

Another innovation of our work is the efficient support of secure monitoring services, which are expected to generate large quantities of data in a continuous way. Related works such as [19] follow a request-response model in which users must pull the network for data. This behavior, if applied to monitoring services, may result in a big overhead for the network and its devices. In contrast, our proposal is based on a secure protocol in which the user engages a monitoring request and, from that point, IoT devices and network nodes (re-)transmit data periodically and asynchronously.

Dealing with security issues in IoT-based services that generate continuous streams of data complicates significantly the security enforcement in comparison with a simple request-response model. Our work brings security to fog-enabled monitoring services while assuming the most challenging scenario in which the entities in the system exchange data in an open network and fog nodes and IoT devices are not fully trusted (i.e., they can be subjected to a wide range of passive and active attacks).

The empirical results we report in Section 6 show that our security-enabling protocols retain the scalability of the system and do not significantly hamper the latency and energy consumption of fog-enabled monitoring.

The rest of the paper is organized as follows. Section 2 describes and dis-

cusses related works. Section 3 presents our approach to enhance security and privacy in fog-enabled IoT monitoring services. Section 4 formalizes the protocols we propose for secure orchestration and secure monitoring. Section 5 analyzes the security and privacy guarantees of our approach. Section 6 analyzes the performance of our solution focusing on the most constrained devices of the network, and comparing the achieved performance with two representative schemes from the literature. Section 7 reports the conclusions and depicts some lines of future work.

2. Related work

Monitoring services bear similarity with the data stream processing scenario, in which stream producers serve data in real-time, and stream consumers gather these data [20]. Security solutions proposed for this scenario are based on data encryption and standard access control models, such as Role-based access control (RBAC) [21, 22, 23, 24, 25, 26], an approach that has been also used in fog-enabled healthcare monitoring [27, 11, 12].

The main limitation of these approaches is that standard access control requires a fully-trusted stream processing infrastructure that stores access policies and enforces access control on sensitive resources. On the one hand, relying on a centralized data stream processing infrastructure goes against the distributed and ubiquitous nature of fog-enabled IoT services. On the other hand, these approaches only offer protection against non-authorized adversaries and consider that authorized IoT devices with proper credentials are fully trusted. Therefore, they are not suitable for scenarios in which authorized IoT devices may be compromised and may misbehave and perform security and privacy attacks.

Attribute-Based Encryption (ABE) [28] offers a more flexible alternative to enforce access control. With cryptographic access control, sensitive resources remain encrypted and only the entities with the appropriate credentials (attributes) will be able to decrypt them. As a result, approaches based on ABE do not require a centralized infrastructure that matches access policies and entity credentials. Due to its inherent advantages, ABE is the most common solution employed to bring security to untrusted, honest-but-curious data stream management systems [29, 30, 31]. Decentralized fog computing architectures also employ ABE as the most common security-enabling solution [32, 33, 34]. ABE fits the needs of IoT-based services because it supports fine-grained access control based on the recipients' attributes, access control is seamlessly enforced by the entities themselves and it is robust against collusion attacks conducted by entities with complementary attributes.

Nevertheless, ABE bears serious limitations when deployed on infrastructures with low tier IoT devices. ABE encryption/decryption primitives are computationally expensive and may introduce a significant overhead for the least capable devices. In particular, employing ABE during the whole service delivery may significantly increase latency [35, 36, 19]. This is specially critical in monitoring services because: i) many services monitor critical signs and, hence, responses should not be subjected to high latencies; and ii) monitoring

services usually involve continuous transmission of data; therefore, the overhead of the (expensive) encryption/decryption operations required by ABE-based access control will be very significant.

To minimize the overhead introduced by ABE in low tier devices, some authors have proposed outsourcing cryptographic operations to a powerful proxy such as the cloud [37]. However, this goes against the philosophy of fog computing, which is to avoid the communication of IoT devices and the cloud.

Homomorphic encryption is another cryptographic solution that has been proposed to tackle security in fog computing [18]. With homomorphic encryption, sensitive data remains encrypted during the whole transmission, and entities with appropriate materials can execute operations on such data without needing to decrypt them. Even though this behavior is very convenient, being significantly costlier than ABE, homomorphic encryption is still far from being practical [38].

Very recently, *fog orchestration* has been proposed as a way to lower the latency of fog-enabled services [15]. Fog orchestration discovers and isolates the network nodes (IoT devices and fog nodes) that are strictly needed to deliver a service, thus saving the resources of the nodes that are not involved in the service and minimizing the bandwidth consumption caused by data broadcasting. Practical research on fog orchestration for IoT is still at its infancy and most works published so far are on the conceptual side [15]. Only [19] proposes practical protocols for fog orchestration with the aim to reduce the cost of cryptographic-based security in fog-enabled services.

Nonetheless, whereas [19] focuses on single request-response services, which can use request-specific session keys, our work tackles monitoring services. The latter are more challenging because data monitoring is a continuous process that may have a long duration and in which data should be served asynchronously. A service with those requirements cannot be efficiently served by means of request-response communications, due to the fact that the overhead introduced by the significant number of messages (and related cryptographic operations), which are required to serve each data update, will severely impair performance.

3. Our proposal

In this section we describe the properties of the scenario we tackle. In particular, we discuss the computational capabilities of the entities involved in the network and the security model.

We consider a hierarchical architecture of fog and IoT nodes, which is the most commonly employed in fog-based monitoring services [11, 12, 10, 39]. As shown in Figure 1, IoT devices (sensors) are the leaves of the hierarchy and can be physically deployed in geographically distributed locations (e.g., sensors worn by patients within hospital facilities or at home). Each device is capable of measuring at least one magnitude (e.g., heart rate, body temperature, etc.) and sending the readings to the immediate upper node in the network. Being energy-constrained low tier devices, they only answer to requests received by the

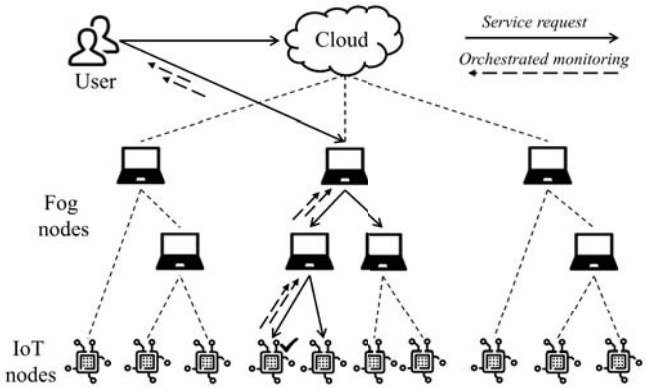

Figure 1: General architecture of a hierarchical fog-based system

fog nodes at upper levels in the hierarchy and are not suited for computationally expensive operations. We consider that sensors may be deployed in untrusted areas (e.g., patients' home) and transmit data through untrusted WAN networks (e.g., the Internet); therefore, they may be subjected to attacks or even become fully controlled by attackers.

Fog nodes, on the other hand, are fully-fledged devices located at the edge of the network that act as gateways between sensors and the cloud. They can be hierarchically organized, e.g., in zones or according to the type of sensing devices they control. In such case, fog nodes are in charge of recursively forwarding user requests and data to/from the sensor nodes. We also consider that fog nodes are untrusted devices that can be subjected to attacks. In the best case, they are expected to behave in an honest-but-curious way, this is, they will follow the protocols but may try to gather users' sensitive data.

The users of the system (e.g., doctors) may monitor one or several sensors according to certain criteria (e.g., their location and/or sensed magnitudes) and with a certain frequency. For this, the user sends her request for a monitoring service to the cloud, and this entity will redirect her to the most appropriate fog node at the edge of the network. This fog node will be the entry point of the user into the fog computing system. Next, the user sends her request for a monitoring service through that entry point to other intermediate fog nodes or sensor nodes at the leaves of the hierarchy (solid arrows in Figure 1). In our proposal, fog nodes orchestrate the network so that only those devices that are strictly needed to fulfill the request will participate in the monitoring (dashed arrows in Figure 1). This avoids data broadcasting and frees the resources of nodes not directly involved in the monitoring. To enforce security, our proposal leverages ABE, symmetric and public cryptography to secure the (sensitive) data streams sent through the network, detect and counter-measure attacks and ensure that only the data that is strictly involved in a monitoring request is revealed to the involved entities, thus fulfilling the *data minimization* principle

included in the EU General Data Protection Regulation (GDPR)[2]. Specifically, -expensive- ABE is used only once during the orchestration step, which requires multicasting the monitoring request in order to discover suitable devices and tailor the network to the request. We also employ the orchestration step to securely create tailored cryptographic materials that will be employed during the monitoring. In this way, the -continuous- transmission and management of sensed data at the monitoring stage are made secure via efficient symmetric cryptography, thereby minimizing the latency and energy consumption of the monitoring.

3.1. Secure orchestration

Orchestrating the network for a monitoring request requires multicasting the request through the nodes in the network. Because fog nodes and sensors are untrusted, this causes security and privacy threats. To tackle them, we employ Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [40] as described below.

At the deployment stage, fog nodes and sensors are given keys associated to attributes that represent their abilities to fulfill the user's request for monitoring. For sensor nodes, the attributes correspond to the magnitudes they sense and/or their locations. For fog nodes, the attributes represent the geographical area they cover and/or the type of devices they control. The idea is that fog node attributes correspond to generalizations of the abilities of the nodes below the fog node in the hierarchy.

More specifically, sensor and fog nodes are given keys associated to the set of attributes that represent their features and also all the generalizations of such features. For example, a sensor capable of measuring the *body_temperature* that is bounded to *patient_X*, may be given a secret key linked to the attributes {*body_temperature*, *vital_sign*, *patient_X*, *room_Y*, *department_Z*}, where *vital_sign* is a generalization of *body_temperature* and *room_Y* and *department_Z* are successive generalizations of the location of *patient_X*.

When a new request for monitoring is received from the user, the orchestration process begins. As a result, the fog nodes and the sensors capable of fulfilling the request are determined, whereas the remaining nodes in the network are pruned. We enforce this by using a *service identifier*, which will be used to generate the temporary cryptographic materials that will be used during the stream monitoring stage. CP-ABE is employed to encrypt the identifier so that only the nodes matching the request (i.e., those having the appropriate attributes) are able to decrypt it and, therefore, participate in the monitoring. The latter brings security, but also privacy to the orchestration, because the user's request is only disclosed to the nodes fulfilling it.

To properly route the request through the hierarchy of nodes, as many policies (stating the set of attributes that the receivers should fulfill) as combinations

[2]http://www.eugdpr.org/

of attribute generalizations of the specific attributes in the request are generated. For example, let us assume that the request consists of monitoring the *body_temperature* of *patient_X* who is located in *room_Y* within *department_Z*. According to the attribute generalizations depicted above, the following policies encompassing all the combinations of attribute values and attribute generalizations will be defined:

- $P_1 = (body\_temperature\ AND\ patient\_X)$
- $P_2 = (vital\_sign\ AND\ patient\_X)$
- $P_3 = (body\_temperature\ AND\ room\_Y)$
- $P_4 = (vital\_sign\ AND\ room\_Y)$
- $P_5 = (body\_temperature\ AND\ department\_Z)$
- $P_6 = (vital\_sign\ AND\ department\_Z)$

The service identifier is encrypted with each of these policies, thereby generating as many ciphertexts as policies. The obtained ciphertexts are sent to the fog node acting as entry point for the user, which will try to decrypt them (see Section 4.1). If at least one is successfully decrypted, then the node has some descendants that are able to fulfill the request. In such case, the remaining ciphertexts, which correspond to policies expressing more specific capabilities, are recursively sent to all the children nodes. If an intermediate node is not able to decrypt any of the ciphertexts it receives, it does nothing, because this means that the branch of nodes below it cannot fulfill the service and they will not participate in the monitoring. Finally, if a sensor at the end of the hierarchy is able to decrypt the ciphertexts it receives, it means that it can handle the request. Next, the selected sensor(s) securely answer the request by providing their *device identifiers*. Intermediate nodes that receive those device identifiers on their way through the user generate pairwise symmetric secret keys bounded to the service identifier and shared with the sensor nodes. The user, upon reception of those device identifiers, is aware of the subset of nodes that will participate in the service delivery; then, she requests the cloud to generate the temporary symmetric keys of the participant sensor nodes bounded to the service identifier. Finally, the sensor nodes are able to generate all the shared cryptographic material by their own. Section 4.1 depicts and formalizes the steps of this protocol.

3.2. Secure monitoring

After the orchestration process, those nodes capable of serving the monitoring request are aware and they have generated all the temporary cryptographic material needed for specifically delivering the requested monitoring procedure. Next, the user can start the monitoring process by multicasting a challenge to the orchestrated network. This challenge contains the identifier of the monitoring process, the starting time and the desired response time (i.e., sensed data

periodicity). The user can re-start the monitoring process anytime by multicasting a new challenge, this may allow her to increase or decrease the response time at will. This challenge is protected by means of cryptography to be strong against integrity and authentication attacks. Moreover, the starting time and the identifier assure its freshness.

Once the orchestrated sensor nodes receive the challenge, they start to sense the environment according to the periodicity indicated by their hardware, and they transmit the retrieved values towards the user in a continuous way. The transmitted data are protected against confidentiality, integrity and authentication attacks by means of symmetric cryptography; and it is in-network aggregated, according to the specified response time by the intermediate fog nodes of the orchestrated hierarchy on its way towards the user. The protection measures added to the transmitted data allow intermediate fog nodes to verify on-the-fly whether the received data have been object of an integrity or authentication attack and close communications with the child node who has sent the affected data. Being capable of detecting and discarding bogus transmissions on-the-fly is essential to deal with continuous streams of data while preserving security.

Finally, the requesting user retrieves the aggregated sensed data, verifies them and, finally, decrypts them. Section 4.2 depicts and formalizes the steps of this protocol.

4. Protocols

This section depicts and formalizes the two protocols introduced above: *Secure orchestration* and *Secure monitoring*. The formal explanation is accompanied by two sequence diagrams (see Figure 2 and 3) depicting the entities involved in the protocols and a simplified view of their interactions (messages). Also, Table 1 summarizes the notation used in both procedures.

In order to run both protocols, the entities that belong to the fog computing system and the users that will use it require storing some cryptographic materials. Those resources are stored in fog and sensor nodes by the organization in charge of the network before being deployed; on the other hand, the users get them by means of a registration step. Whatever the case is, we next summarize the cryptographic materials that each entity must own before following the *Secure orchestration* protocol:

- *Cloud*: It stores all the symmetric secret keys shared with all the fog nodes and sensor nodes; the public key PK^{cp-abe} and master key MK^{cp-abe} of the CP-ABE cryptosystem in use; and the CP-ABE keys assigned to each node.
- *Fog node (FN_i)*: It stores a symmetric secret key MK_{FN_i} shared with the cloud; a public/private key pair (i.e., PK_{FN_i}/SK_{FN_i}) from a public-key cryptosystem with signing capabilities (e.g., DSA/ECDSA[3]), in which

[3]http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

Table 1: Notation

Notation	Description
U	User that uses the proposed system to run a monitoring service
v	Random value that identifies the monitoring service that U is willing to run
FN_i	Fog node with identifier i
FN_{ep}	Fog node that will behave as entry point of U into the fog computing system
SN_i	Sensor node with identifier i
ID_{X_i}	Identifier in the fog computing system of node X_i
$GATR_U$	Set of the most generalized attributes that best describe the monitoring service requested by U
ATR_U	Set of the specific attributes that describe the monitoring service requested by U
MK^{cp-abe}	Master key owned by the cloud and generated from the CP-ABE cryptosystem in use
PK^{cp-abe}	Public key linked to MK^{cp-abe}
$SK^{cp-abe}_{X_i}$	CP-ABE secret key owned by node X_i, linked to its capabilities, and generated from MK^{cp-abe}
P_x	Policy stating the Boolean combination of attributes required to decrypt a CP-ABE ciphertext
CT_x	CP-ABE ciphertext generated from PK^{cp-abe} that encrypts v under a policy P_x
MK_{X_i}	Symmetric secret key shared between node X_i and the cloud
SK_{FN_i}	Secret key owned by FN_i and generated from a public-key cryptosystem with signing capabilities
PK_{FN_i}	Public key from a public-key linked to SK_{FN_i}
$K^v_{X_i}$	Temporal symmetric secret key linked to service v and shared between node X_i and U (and the cloud)
$K_{SN_i-FN_i}$	Pairwise symmetric secret key between SN_i and FN_i
$K^v_{SN_i-FN_i}$	Temporal version of $K_{SN_i-FN_i}$ linked to service v
t_U	Starting time of the monitoring service run by U
t_{rt}	Response time in which the network has to feed periodically with new sensor readings
t_{SN_i}	Time of the current sensor reading
$(b_1, \ldots, b_t)$	Binary representation of the sensor reading to be transmitted by SN_i
$(c_1, \ldots, c_t)$	Pseudo-random sequence used to encrypt $(b_1, \ldots, b_t)$
$(d_1, \ldots, d_t)$	Encrypted version of $(b_1, \ldots, b_t)$
σ_X	HMAC to be verified by node X
σ'_X	Value that aggregates all the HMACs to be verified by node X
ω	Value computed by node X that is compared with a received σ'_X

PK_{FN_i} is accepted as valid by all nodes located in the subtree rooted by FN_i; and a CP-ABE secret key $SK^{cp-abe}_{FN_i}$ related to its capabilities.

- *Sensor node (SN_i)*: It stores a symmetric secret key MK_{SN_i} shared with the cloud; the public keys PK_{FN_i} of the fog nodes that are situated in the path towards the cloud; and a CP-ABE secret key $SK^{cp-abe}_{SN_i}$ related to its capabilities.

- *User*: She stores a digital certificate to authenticate herself in front of the cloud; the universe of attributes that define the capabilities of the fog nodes and sensor nodes that form the fog computing system; and the public key PK^{cp-abe} linked to the CP-ABE cryptosystem.

4.1. Protocol-1: Secure orchestration

4.1.1. User joins the system

A user U enters the fog computing system as follows:

a) U establishes a SSL/TLS secured communication with the cloud and authenticates herself.

b) The cloud verifies that U is allowed to use the fog computing system.

c) U generates a fresh random value v (freshness can be guaranteed by adding a timestamp), which will identify the monitoring service that U will execute. U also uses the universe of attributes linked to the capabilities of the fog/sensor nodes to define the most generalized set of attributes $GATR_U$ that best describe the requested monitoring (e.g., those used in P_6 in the request example in Section 3.1).

d) U sends the pair $\{v, GATR_U\}$ to the cloud.

e) The cloud uses $GATR_U$ to decide which fog node FN_i is best suited to handle the requested monitoring. In this way, the selected FN_i will be the entry point of U into the fog computing system. In the following, we label this entry point fog node as FN_{ep}.

f) The cloud checks that v is a fresh value (it has not been used previously) and builds the symmetric key $K^v_{FN_{ep}} = H(v||MK_{FN_{ep}})$ that will be shared between FN_{ep} and U during the monitoring process. Note that $||$ is the concatenation operator.

g) The cloud sends the pair $\{URI_{FN_{ep}}, K^v_{FN_{ep}}\}$ to U. Where $URI_{FN_{ep}}$ is a uniform resource identifier that may allow U to contact FN_{ep} remotely.

4.1.2. User orchestrates a hierarchy of nodes to run the monitoring service v

A user U, who has already joined the system, orchestrates a subset of nodes of the fog computing network to run the monitoring service identified by the value v generated at the former step. This is done as follows:

a) U defines the specific attributes ATR_U of her service and builds as many policies $(P_1, \ldots, P_n)$ as combinations of (generalized) attribute types.

b) U generates as many CP-ABE ciphertexts $(CT_1, \ldots, CT_N)$ as policies by computing:

$$CT_x = Encrypt(PK^{cp-abe}, v, P_x)$$

c) U builds a message containing the CP-ABE ciphertexts and protected by a HMAC, and sends it to its assigned entry point FN_{ep}:

$$\{(CT_1, \ldots, CT_N), HMAC((CT_1, \ldots, CT_N), K^v_{FN_{ep}})\}$$

d) FN_{ep} tries to decrypt $(CT_1, \ldots, CT_N)$ using its $SK_{FN_{ep}}^{cp-abe}$, which is associated to the (generalized) attributes that encompass the capabilities of the nodes below FN_{ep} in the hierarchy; that is $Decrypt(PK^{cp-abe}, CT_x, SK_{FN_{ep}}^{cp-abe})$.

- If at least one of the CP-ABE ciphertexts is successfully decrypted, FN_{ep} gets v and verifies the HMAC. Next, FN_{ep} multicasts to its descendants in the hierarchy i) its identifier in the fog computing system and ii) the list of non-decrypted CP-ABE ciphertexts, together with the corresponding digital signatures of each of those elements built by means of its private key $SK_{FN_{ep}}$. This is:

$$\{ID_{FN_{ep}}, (CT_x, \ldots, CT_N), (ID_{FN_{ep}})_{SK_{FN_{ep}}}, \\ (CT_x)_{SK_{FN_{ep}}}, \ldots, (CT_N)_{SK_{FN_{ep}}}\}$$

- Else, FN_{ep} does not continue the protocol.

e) Intermediate fog nodes FN_i receive the message above and they first verify the digital signatures; next, they try to decrypt the CP-ABE ciphertexts using their CP-ABE secret keys.

- If at least one of the CP-ABE ciphertexts is successfully decrypted, FN_i gets v and forwards the remaining non-decrypted CP-ABE ciphertexts and their digital signatures to its descendants, adding to this message its own identifier in the fog computing system (i.e., ID_{FN_i}) and its corresponding digital signature. This is:

$$\{ID_{FN_i}, ID_{FN_{ep}}, (CT_y, \ldots, CT_N), \\ (ID_{FN_i})_{SK_{FN_i}}, (ID_{FN_{ep}})_{SK_{FN_{ep}}}, \\ (CT_y)_{SK_{FN_{ep}}}, \ldots, (CT_N)_{SK_{FN_{ep}}}\}$$

  This process is repeated until the leaves of the hierarchy (i.e., the sensor nodes) are reached. The list of node identifiers received by the leaves will represent the *path* of fog nodes between the sensor node and the user U.

- Else, FN_i does not continue the protocol and the subtree of nodes rooted at this entity is pruned by the orchestration process.

f) Sensor nodes SN_i receive the CP-ABE ciphertexts and the *path* of fog nodes towards U. They first verify the digital signatures; next, they try to decrypt the CP-ABE ciphertexts using their CP-ABE secret keys.

- If at least one of the CP-ABE ciphertexts is successfully decrypted, SN_i gets v and it will participate in the requested monitoring service by sensing and transmitting its readings. This is:

i) SN_i builds a temporary symmetric secret key bounded to service v to be shared with U:

$$K^v_{SN_i} = H(v||MK_{SN_i})$$

ii) SN_i builds symmetric secret keys bounded to the monitoring service v to be shared with each fog node FN_i situated in its path towards U. To do that, SN_i applies a pairwise key establishment scheme such as the Blom's scheme [41]. By means of this approach, SN_i can use the received identifier of each fog node (e.g., ID_{FN_i}) and its own master secret key MK_{SN_i} to generate a pairwise symmetric key $K_{SN_i-FN_i}$ with each fog node in that path: $K_{SN_i-FN_i} = (MK_{SN_i})^T \cdot ID_{FN_i}$, where T denotes matrix transpose.
Next, SN_i computes a temporary version of each pairwise key bounded to service v:

$$K^v_{SN_i-FN_i} = H(v||K_{SN_i-FN_i})$$

These temporary keys will be used to perform the *Secure monitoring* protocol (see Section 4.2).

iii) SN_i builds a message containing: a hash of the service identifier v, its identifier in the fog computing system, and the set of identifiers of the fog nodes who form the path towards U. This message is protected by means of an HMAC built using $K^v_{SN_i}$:

$$\{H(v), (ID_{SN_i}, ID_{FN_i}, \ldots, ID_{FN_{ep}}),$$
$$HMAC(v||ID_{SN_i}||ID_{FN_i}||\ldots||ID_{FN_{ep}}, K^v_{SN_i})\}$$

Note that $H(v)$ allows both the user and the fog nodes to identify (without disclosing v) messages belonging to a certain orchestration procedure among other possible orchestration procedures being tackled concurrently.

iv) SN_i sends this message towards U by means of its ancestor in the hierarchy. The message will be forwarded up to the top of the tree until FN_{ep} is reached.

- Else, SN_i does not continue the protocol and it is pruned by the orchestration process.

g) Each fog node that receives the message sent by SN_i towards U retrieves the identifier of SN_i (i.e., ID_{SN_i}) and computes its pairwise symmetric secret key bounded to service v and shared with SN_i. Again, applying the Blom's scheme [41], it could be done by first computing $K_{SN_i-FN_i} = (MK_{FN_i})^T \cdot ID_{SN_i}$; and, finally, $K^v_{SN_i-FN_i} = H(v||K_{SN_i-FN_i})$. These

temporary keys will be used to perform the *Secure monitoring* protocol (see Section 4.2).

Then, SN_i's message is forwarded to the next ancestor on its way up to U.

h) U receives messages containing the identifiers of all the sensor nodes that will participate in the requested monitoring and the identifiers of the paths of fog nodes that will deliver their readings. Then, U sends value v and those paths of nodes to the cloud.

i) The cloud performs two operations:

- It uses its knowledge of the topology of the fog computing network to verify whether the paths of fog and sensor nodes received by U are correct or not. If they are not correct, the cloud may run a protocol for suppressing compromised nodes from the network.

  Selecting a suitable protocol for suppressing compromised nodes is out of the scope of this paper, but a procedure which can be found in the literature and that would be suitable for the considered scenario is presented in [42].
- It uses the sensor nodes' identifiers to retrieve their symmetric secret keys and, by means of v, it computes their temporary secret keys $K^v_{SN_i} = H(v||MK_{SN_i})$. Then, the cloud sends them to U.

j) U uses keys $K^v_{SN_i}$ to verify the HMACs send by the sensor nodes. If the verification succeeds, U will use these temporary secret keys to perform the *Secure monitoring* protocol (see Section 4.2).

If the verification fails, U will inform the cloud about this situation and will provide the HMACs as proofs. The cloud, then, may run a protocol for suppressing compromised nodes from the network, as indicated previously.

4.2. Protocol-2: Secure monitoring

The objective of this protocol is to perform the *Secure monitoring* process (see Section 3.2) by means of the subset of nodes that have been orchestrated in the former protocol.

The type of monitoring services that we envisage in this paper are assumed to retrieve sensed data from the sensor nodes codified as a binary sequence. When challenged by the user, sensor nodes will continuously send their readings, according to the periodicity indicated by its hardware. On the other hand, intermediate fog nodes will aggregate those sensor readings and they will send the result towards the user according to the time periodicity indicated by her.

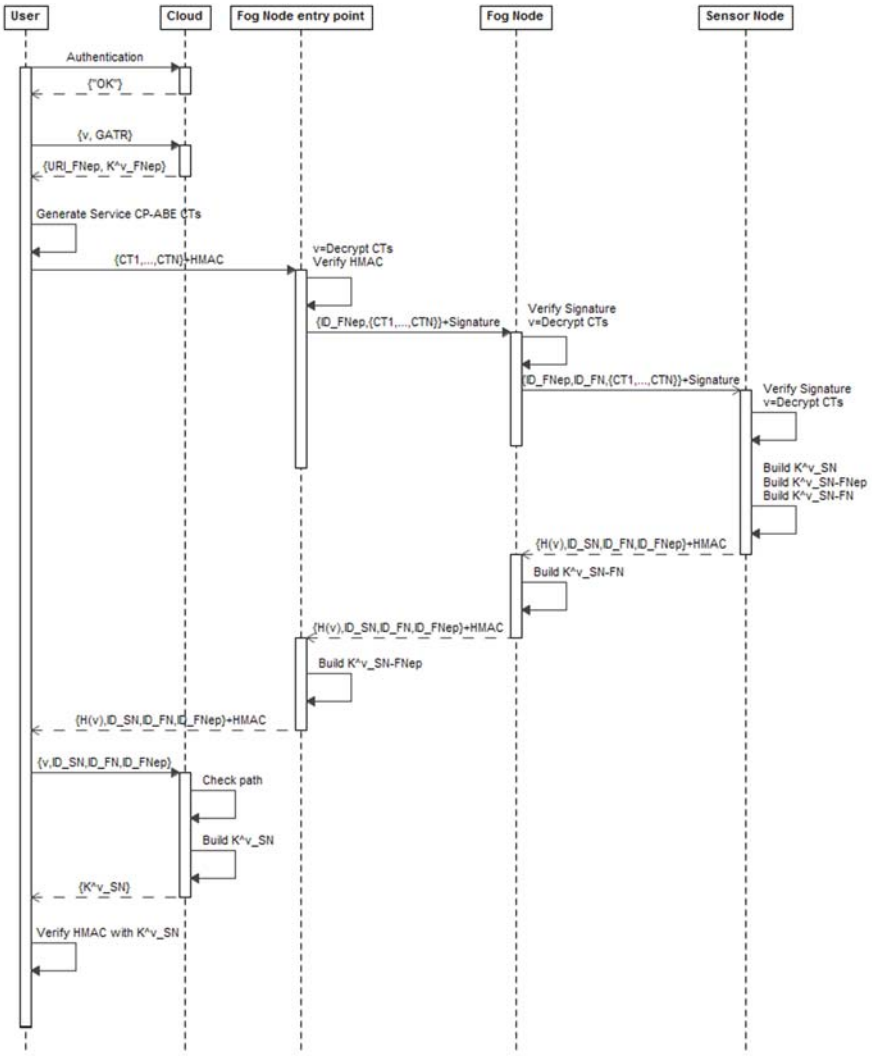

Figure 2: Sequence diagram describing the entities and messages involved in Protocol-1

4.2.1. User starts the monitoring service

User U challenges the orchestrated subset to start (or re-start) the sensing and delivery of the data. This is done as follows:

a) U obtains the current time t_U, which indicates the starting time of the monitoring service identified by v. U also specifies the response time t_{rt} in which the network has to feed her periodically with new sensor readings.

b) U builds a message containing these data and protected by a HMAC, and sends it to its assigned entry point FN_{ep}:

$$\{v, t_U, t_{rt}, HMAC(v||t_U||t_{rt}, K^v_{FN_{ep}})\}$$

c) FN_{ep} receives that message and performs the following steps:

   i) It verifies the HMAC. If this verification fails, the message is discarded and FN_{ep} does not continue the protocol. Else, FN_{ep} continues with the following steps.

   ii) It verifies whether t_U is greater than the previous most recent starting time sent by U for service v; this will prove the freshness of the U's request. If this verification fails, the message is discarded and FN_{ep} does not continue the protocol. Else, FN_{ep} continues with the following steps.

   iii) It stores t_U as the most recent starting time sent by U for service v. It also sets t_{rt} as the current response time for sending new sensor reading periodically to U.

   iv) It signs $\{v, t_U, t_{rt}\}$ using its secret key $SK_{FN_{ep}}$, and multicasts the resulting message to its descendants in the hierarchy.

d) A fog node FN_i that receives $(v, t_U, t_{rt})_{SK_{FN_{ep}}}$ performs the following operations:

   i) It verifies the signature. If this verification fails, the message is discarded and FN_i does not continue the protocol. Else, FN_i continues with the following steps.

   ii) It verifies whether t_U is greater than the previous most recent starting time sent by U for service v. If this verification fails, the message is discarded and FN_i does not continue the protocol. Else, FN_i continues with the following steps.

   iii) It stores t_U as the most recent starting time sent by U for monitoring v. It also sets t_{rt} as the current response time for sending new sensor readings periodically to its ancestor in the orchestrated network.

   iv) It forwards the received data to its descendants. This is done iteratively until all the leaves of the orchestrated hierarchy (the sensor nodes) have received the data.

e) A sensor node SN_i that receives $(v, t_U, t_{rt})_{SK_{FN_{ep}}}$ performs the following operations:

i) It verifies the signature. If this verification fails, the message is discarded and SN_i does not continue the protocol. Else, SN_i continues with the following steps.

ii) It verifies whether t_U is greater than the previous most recent starting time sent by U for service v. If this verification fails, the message is discarded and SN_i does not continue the protocol. Else, SN_i continues with the following and final step.

iii) It stores t_U as the most recent starting time sent by U for service v. Value t_{rt} is discarded since sensor nodes are assumed to gather sensed data and sends them to their ancestors in the orchestrated network at the periodicity indicated by their hardware.

4.2.2. Sensor nodes iteratively send the sensed data towards the user via the orchestrated hierarchy

Due to the orchestration process already performed and the reception of U's monitoring service challenge, at this point, a sensor node SN_i which is ready to read and send data knows:

- A symmetric secret key $K^v_{SN_i}$ bounded to service v and shared with U.
- The symmetric secret keys $K^v_{SN_i-FN_j}$ bounded to service v and shared with each fog node FN_j situated in its path towards U.
- The most recent starting time t_U sent by U for service v.

Let $(b_1, \ldots, b_t)$ be the binary representation of the sensor reading to be transmitted by SN_i. Upon reception of the challenge, SN_i continuously sends (i.e., according to the periodicity indicated by its hardware) its readings to U by following the next steps:

a) It gets the time t_{SN_i} of the current sensor reading by means of SN_i's internal clock and the external time t_U.

b) It computes a pseudo-random t-bit sequence:

$$(c_1, \ldots, c_t) \leftarrow lsb_t(H(v||t_{SN_i}||K^v_{SN_i}))$$

where $lsb_t(\cdot)$ is a function returning the t least significant bits of its argument.

c) It encrypts $(b_1, \ldots, b_t)$ in a sequence $(d_1, \ldots, d_t)$ as follows:

$$(d_1, \ldots, d_t) \leftarrow (b_1 \oplus c_1, \ldots, b_t \oplus c_t)$$

d) It generates a HMAC of $(d_1, \ldots, d_t)$ to be verified by U; and a HMAC of the same data to be verified by each fog node FN_j situated in its path towards U:

$$\sigma_U \leftarrow HMAC(d_1, \ldots, d_t||v||t_{SN_i}, K^v_{SN_i})$$
$$\sigma_{FN_j} \leftarrow HMAC(d_1, \ldots, d_t||v||t_{SN_i}, K^v_{SN_i - FN_j})$$

e) It sends up to its ancestor in the orchestrated hierarchy the following message:

$$\{ID_{SN_i}, (d_1, \ldots, d_t), t_{SN_i}, \sigma_U, \sigma_{FN_a}, \ldots, \sigma_{FN_z}\}$$

4.2.3. Intermediate nodes aggregate the data in transit

Fog node FN_i receives messages sent by each of its descendants (i.e., sensor nodes or other fog nodes). Depending on its position on the orchestrated hierarchy, each received message may contain data from only one sensor node or from m different ones. Those messages have the following format:

$$\{(ID_{SN_a}, (d_{1,a}, \ldots, d_{t,a}), t_{SN_a}), \ldots, (ID_{SN_z}, (d_{1,z}, \ldots, d_{t,z}),$$
$$t_{SN_z}), \sigma_U, \sigma_{FN_l}, \ldots, \sigma_{FN_k}\}$$

FN_i aggregates the received messages and forwards the resulting one towards U by following these steps:

a) FN_i continuously receives messages from its descendants in the orchestrated hierarchy and, for each reception, performs the following operations:

i) It keeps track of the latest t_{SN_i} for each SN_i, and uses this knowledge to verify the freshness of the received message. If this verification fails, FN_i marks the transmitting descendant as untrusted and it does not accept future transmissions from this source. Else, FN_i continues with the following steps.

ii) It gets the data from the m sensor nodes involved in that specific message, and computes its own aggregated HMAC value as follows:

$$\omega_{FN_i} = \sum_m HMAC(d_{1,m}, \ldots, d_{t,m}||v||t_{SN_m}||K^v_{SN_m - FN_i}) \pmod{2^{HMAC\_bitlength}}$$

iii) It gets σ_{FN_i} from the received message, and checks whether $\sigma_{FN_i} = \omega_{FN_i}$.

- If this verification fails, first, FN_i marks the transmitting descendant as untrusted and it does not accept future transmissions from this source. Next, FN_i informs the cloud about this incident. The cloud can then perform a revision process of the affected nodes that may include a physical analysis done by a field technician. The details of such a revision process are beyond the scope of this paper.
- Else, FN_i eliminates component σ_{FN_i} from the received message and stores the rest of it in a temporary memory overwriting the former transmission sent by the same descendant; also, it updates the latest t_{SN_i}.

b) According to the periodicity indicated by the response time t_{rt}, FN_i aggregates the messages from all the active descendants which are stored in the temporary memory by performing the following operations:

i) It concatenates the identifiers, symbols and sensing times present in all the stored messages:

$$(ID_{SN_a}, (d_{1,a}, \ldots, d_{t,a}), t_{SN_a}), \ldots, (ID_{SN_z}, (d_{1,z}, \ldots, d_{t,z}), t_{SN_z})$$

ii) It checks the stored messages and aggregates all the σ values linked to U and the σ values linked each fog node FN_j found:

$$\sigma'_U = \sum_m \sigma_U \pmod{2^{HMAC\_bitlength}}$$
$$\sigma'_{FN_j} = \sum_k \sigma_{FN_j} \pmod{2^{HMAC\_bitlength}}$$

iii) It uses those three aggregated elements to build the final aggregated message:

$$\{(ID_{SN_a}, (d_{1,a}, \ldots, d_{t,a}), t_{SN_a}), \ldots, (ID_{SN_z}, (d_{1,z}, \ldots, d_{t,z}), t_{SN_z}), \sigma'_U, \sigma'_{FN_l}, \ldots, \sigma'_{FN_k}\}$$

c) FN_i sends the final aggregated message to its parent node in the orchestrated hierarchy.

- If the parent node is U (thus, $FN_i = FN_{ep}$), the sensed data have already reached the user, who now will retrieve it.
- Else, this protocol step must be executed again at the parent node (another fog node).

4.2.4. User obtains the sensed data

U receives a final message containing all the readings from all the orchestrated sensor nodes, together with the final aggregated σ''_U:

$$\{(ID_{SN_a}, (d_{1,a}, \ldots, d_{t,a}), t_{SN_a}), \ldots,$$
$$(ID_{SN_z}, (d_{1,z}, \ldots, d_{t,z}), t_{SN_z}), \sigma''_U\}$$

From this message, U obtains the sensed data by following these steps:

a) She computes the pseudo-random t-bit sequence $(c_{1,i}, \ldots, c_{t,i})$ that each SN_i built at Step b) to encrypt their sensor readings. For this purpose, U uses the shared secret keys $K^v_{SN_i}$, the time of the corresponding sensor reading t_{SN_i}, and the random value v that identifies the monitoring service.

b) She retrieves the binary representation of the sensor reading sent by each SN_i as follows:

$$(b_{1,i}, \ldots, b_{t,i}) \leftarrow (d_{1,i} \oplus c_{1,i}, \ldots, d_{t,i} \oplus c_{t,i})$$

c) She computes the HMAC values that correspond to each sensor reading sent by each SN_i, and aggregates all the computed HMACs as follows:

$$\omega = \sum_m (d_{1,m}, \ldots, d_{t,m} || v || t_{SN_i} || K^v_{SN_i})$$
$$(\text{mod } 2^{HMAC\_bitlength})$$

d) She checks whether $\omega = \sigma''_U$.

- If the equality succeeds, U accepts the received sensor readings.
- Else, U does not accept the received symbols and informs the cloud about this incidence. The cloud can then perform a revision process of the affected nodes that may include a physical analysis done by a field technician. The details of such a revision process are beyond the scope of this paper.

5. Security analysis

This section studies the security and privacy capabilities of the proposed system by means of four propositions. Each proposition is supported by a set of claims.

The attack model considered is the same that in [19]: adversaries can compromise both fog and sensor nodes; they also can access communication lines to capture, modify and resend messages; finally, all non-compromised entities of the system are considered to be *honest-but-curious*, this is, they follow the proposed protocol as expected, but they may try to learn the users' sensitive data if they have the chance.

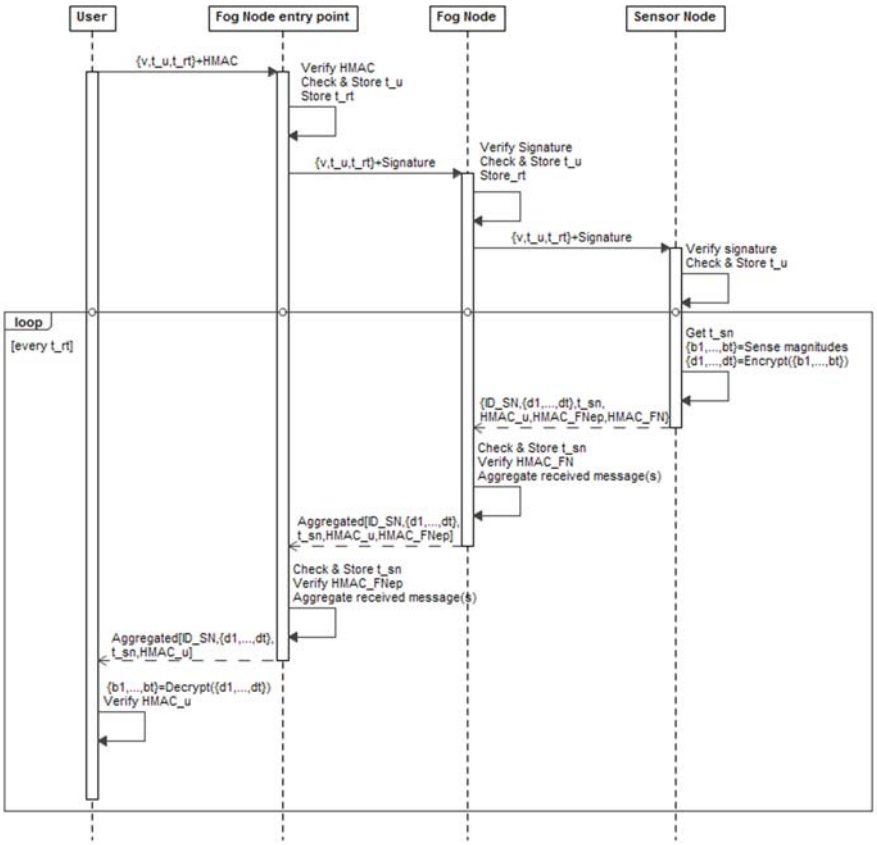

Figure 3: Sequence diagram describing the entities and messages involved in Protocol-2

5.1. Proposition-1: Secure transmission of sensor readings

The new proposal protects the confidentiality, authenticity and integrity of the transmitted sensor readings in front of external or internal attackers that may perform active or passive attacks. If internal attackers control compromised fog or sensor nodes, their attacks will only leak the information known by those affected nodes and they will not perturb the rest of the network. If internal attackers try to perform integrity attacks or refuse to aggregate the data in transit, the proposed system will be capable of detecting this situation. This proposition is supported by five claims.

Claim 1. *An adversary cannot successfully obtain the transmitted sensor readings due to message eavesdropping during the secure monitoring protocol.*

U starts the monitoring service by multicasting in clear the random value v that identifies the service, the starting time of the service t_U and the expected response time t_{rt}. These values provide no useful knowledge to external or internal attackers. As a response to that challenge, the sensor readings are sent towards U and they area aggregated at intermediate fog nodes.

Sensor readings are encrypted at origin by means of a secure fixed-length encryption approach that employs a collision-resistant hash function as pseudorandom generator [43] to perform the encryption of the plaintext. As proved in [44], if the pseudorandom generator is valid, this construction provides indistinguishable encryptions in the presence of an eavesdropper.

The employed encryption approach uses, as inputs for the hash function, temporary secret keys $K^v_{SN_i}$ only known by U and by each sensor node SN_i. A key $K^v_{SN_i}$ is bounded specifically to service v and it is built by means of the corresponding master secret key MK_{SN_i}. Only the specific sensor node SN_i and the trustworthy cloud knows MK_{SN_i}; this data is not disclosed at any protocol step.

In summary, the encryption scheme is secure in the presence of eavesdroppers and external attackers are not capable of obtaining a valid $K^v_{SN_i}$, hence, they cannot get any transmitted sensor reading. Regarding internal attackers, they can only get the $K^v_{SN_i}$ of the sensor nodes that they have compromised; therefore, they can only decrypt and know the sensor readings generated by her own controlled sensor nodes. This issue does not affect the rest of the network.

Claim 2. *An adversary cannot successfully impersonate a certain fog or sensor node in front of other entities.*

Fog and sensor nodes use a temporary secret key $K^v_{FN_i}/K^v_{SN_i}$ to build the HMACs of the messages they share with the users. HMAC is a specific instantiation of the hash-and-MAC paradigm. It is an industry standard and is widely used in practice. According to [44], this construction is supported by a proof of security based on assumptions that are believed to hold for practical hash functions. As a consequence, it can be assumed that, currently, it is computationally unfeasible for adversaries to tamper with HMACs.

The keys used to build the HMACs are computed by means of the corresponding master keys MK_{FN_i}/MK_{SN_i}. Only the specific node FN_i/SN_i and

the trustworthy cloud know MK_{FN_i}/MK_{SN_i}. This data is not disclosed at any protocol step and, hence, an external adversary cannot get access to the required cryptographic material. Regarding an internal adversary, compromising a certain node will not give her any valid information about other nodes, therefore, the system is safe against this kind of attack too.

Focusing on the messages that fog nodes multicast to the sensor nodes, they are digitally signed using secret keys SK_{FN_i} that correspond to a public-key algorithm with signing capabilities (i.e., DSA/ECDSA). DSA/ECDSA are constructions based on the discrete-logarithm problem in different classes of groups, and they are included in the current Digital Signature Standard (DSS) issued by NIST[4]. Currently, there are no proofs that DSA/ECDSA are provable secure, however, both DSA and ECDSA have been used and studied for decades without any attacks being found [44]. In this way, they can be assumed to be secure provided that the signer uses a proper random nonce to generate the corresponding digital signature. Regarding the security of the signing key, it is only known by each fog node and, therefore, an adversary cannot retrieve it.

Claim 3. *An adversary cannot modify the messages sent by other parties without being detected.*

The proposed system uses HMACs or digital signatures (i.e., DSA/ECDSA) to protect all the transmissions between entities.

As it has been previously justified, nowadays, it is computationally unfeasible for adversaries to tamper with HMACs. Regarding DSA/ECDSA, it has been already explained that both constructions are assumed to be secure provided that certain precautions are taken when performing the signing operation. The secret keys used for computing HMACs are only known by the trustworthy cloud and by each node individually. The secret keys used for computing digital signatures are only known by each fog node individually.

As a result, any external or internal adversary trying to modify any transmission will fully disrupt it, but legit entities will detect this situation during the integrity verifications that they perform and will inform the cloud about this incident. Next, the organization that supports the cloud may perform a revision process of the affected nodes that may include a physical analysis done by a field technician.

Claim 4. *An adversary cannot remove the sensor reading sent by a certain sensor node without being detected.*

In order to erase the sensor reading sent by a certain sensor node SN_i, an adversary needs to know the corresponding HMAC σ_i linked to that sensor reading. If the adversary does not know this value, she may guess the correct σ_i with a probability equivalent to the bit length of the HMAC algorithm. On the other hand, if the adversary knows σ_i and she successfully erases the sensor reading, U will detect this situation as a non-transmittal. Due to the *Secure*

[4]https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

orchestration protocol run by U before starting the *Secure monitoring* protocol, U is aware of all the sensor nodes that should send their readings. As a result, a non-transmittal from a certain sensor node will be acknowledged as a sensor reading removal by means of an adversary. This incident will be reported to the cloud which may then perform a revision process of the affected nodes that may include a physical analysis done by a field technician.

Claim 5. *An adversary controlling a compromised fog node cannot refuse to aggregate the sensor readings in transit without being detected.*

A fog node that refuses to aggregate messages coming from its descendants will generate a non-transmittal situation. Similarly to Claim 4, a non-transmittal from a certain node or a set of nodes will be acknowledged by U as an attack from an adversary, and U will be able to ascertain which part of the network has been affected. This incident will be reported to the cloud which may then perform a revision process of the affected nodes that may include a physical analysis done by a field technician.

5.2. Proposition-2: integrity and authenticity attacks detected during the monitoring service are addressed on the fly

The proposed system offers security in monitoring services which, inherently, work with continuous streams of data. Therefore, during the continuous transmission of sensor readings towards U, intermediate fog nodes must be capable of detecting integrity and authenticity attacks performed by external or internal attackers located at lower layers. Also, they must be capable of eliminating the source of the corrupted data. This proposition is supported by one claim.

Claim 6. *An intermediate fog node that receives sensor readings with an incorrect HMAC value from one of its descendants, is capable of detecting this situation and closing further communications with that child; in this way, the affected branch of the hierarchy is pruned.*

During the orchestration protocol, those sensor and fog nodes that are orchestrated generate shared secret keys $K^v_{SN_m-FN_i}$. Those keys are used during the monitoring protocol by the sensor nodes to build HMACs for each fog node situated in the path towards U. In this way, each intermediate fog node is capable of checking its linked HMAC and verifying the integrity and authenticity of the received data. Upon detecting an attack, the fog node is able to close connections with the source of the corrupted message and ignore further incoming transmissions from the pruned descendant.

As it has been previously justified, nowadays, it is computationally unfeasible for adversaries to tamper with HMACs. Regarding the security of the shared secret keys $K^v_{SN_m-FN_i}$, these keys are built by means of a pairwise key establishment scheme (such as the Blom's scheme [37]). Therefore, it is the strength of the selected pairwise key establishment scheme what determines whether an adversary may be able to compute a valid shared key and attack the system. The security of the Blom's scheme and other related schemes directly depends

on the so-called "capture threshold", which is the number of nodes of the system that must be captured, and their cryptographic material retrieved, in order to be capable of computing the shared keys of the rest of the nodes. This threshold depends on a parameter used in the scheme that directly affects the quantity of cryptographic material that each node must store. In this way, the authors in [45] study a variation of the Blom's scheme and they conclude that, with suitable keying parameters, and keeping the stored data affordable for lightweight sensor nodes, the number of captured nodes needed to perform a successful attack is very large and this fact makes this possibility unlikely.

5.3. Proposition-3: only authenticated users can use the proposed system

A user must be properly authenticated to use the proposed system. Adversaries without the proper credentials cannot decrypt the sensor readings even if they control fog nodes. This proposition is supported by two claims.

Claim 7. *In order to get the temporary secret keys $K^v_{SN_i}$ required to decrypt the sensor readings transmitted by the sensor nodes, any user must first authenticate herself in front of the cloud by means of a valid digital certificate.*

An adversary who is willing to use the system without the owning proper rights must be able to get a fake but valid authentication certificate that grants her access. A digital certificate is simply a digital signature binding an entity to some public key. Typical X.509 certificates[5] use algorithms such as RSA, DSA or ECDSA to compute the signature. The security provided by DSA/ECDSA has been already discussed in former claims. Regarding RSA, as it is justified in [44], the RSA full-domain hash (RSA-FDH) signature scheme is considered to be secure. As a result, generating a valid fake credential without knowing the certification authority's secret key is unfeasible.

Secret keys $K^v_{SN_i}$ can only be computed by the cloud and, individually, by each sensor node, hence, compromising a fog node will not make any difference; moreover, compromising a certain sensor node will not give the adversary any valid information about other nodes.

Claim 8. *An adversary cannot reuse past transmissions to repeat the execution of a past monitoring service.*

All transmissions between users and fog and sensor nodes are protected by means of HMAC values. As it has been previously justified, nowadays, it is computationally unfeasible for adversaries to tamper with HMACs. During the orchestration protocol, HMACs are built using a random and fresh value v that identifies the monitoring service. During the monitoring protocol, HMACs are built using v and timestamps (t_U indicates the starting time of the monitoring, t_{SN_i} indicates the time of the last sensor reading generated by SN_i). All nodes keep track of v and the latest times involved during the protocols run; they also verify the HMACs before accepting the received messages. As a result, any reused message containing outdated HMACs will be detected and discarded.

[5] https://tools.ietf.org/html/rfc5280

5.4. Proposition-4: privacy protection by means of the data minimization principle

The new proposal protects the privacy of the users by fulfilling the data minimization principle included in the EU GDPR. More specifically, this principle states that only the data which are strictly necessary to perform a certain monitoring service will be revealed to those entities in charge of the operation. This proposition is supported by two claims.

Claim 9. *An adversary, being external or internal, cannot know the detailed interests of a certain user (i.e., type of data in which she is interested).*

During the orchestration protocol, users send a set of CP-ABE ciphertexts encrypted with the attributes linked to the type of sensed data that they wish to gather. Those encrypted attributes follow a tree model (from more general to more specific) that is mapped to the orchestrated hierarchy of devices. In this way, fog nodes are situated at upper layers and, hence, they are linked to general attributes; on the other hand, sensor nodes are situated at the bottom of the hierarchy and, hence, they are linked to specific attributes.

The security model of the CP-ABE cryptosystem introduced in [40] indicates that it can handle chosen-plaintext attacks. The authors also argue that this construction can easily be extended to handle chosen-ciphertext attacks as well. Moreover, all entities can use the public key of the CP-ABE cryptosystem to compute those CP-ABE ciphertexts, but only the nodes that know the right secret keys are able to decrypt them and, hence, ascertain whether their attributes fit with the attributes of the requested service.

An external adversary will not be able to break the CP-ABE cryptosystem and she will not have the required cryptographic material either. As a result, she will not be able to decrypt the CP-ABE ciphertexts and get the interests of the users. Regarding an internal adversary, the amount of information that she will be able to retrieve using her secret keys will depend on the hierarchy layer in which the compromised devices are located.

During the monitoring protocol, only those sensor nodes that have been orchestrated according to their attributes will send sensed data towards U. An adversary who knows the specific attributes of the sensors that transmit data can, then, infer U's interests; however, this situation is only plausible and effective in the case that this adversary has this knowledge for a big part of the sensors involved in a specific service.

Claim 10. *Users can only obtain sensor readings up to the level of detail to which they are authorized.*

As indicated in the previous claim, sensor nodes (and their sensed data) are linked to specific attributes that may provide detailed personal information. A user U, in order to run a certain monitoring service, must first authenticate herself in front of the cloud and provide a set of attributes $GATR_U$ that describes the requested service. The level of detail required for $GATR_U$ for a certain service will be indicated by the cloud, and it may reject a service request based on

U's identity and whether she is authorized to run a service with the indicated type of attributes.

6. Performance analysis

The proposed system is made of two protocols: *Secure orchestration* and *Secure monitoring*. Both run in an architecture made of fog nodes, sensor nodes and the cloud. Sensor nodes are battery-powered lightweight devices and, for the system to be scalable, these devices must run protocol operations that involve: i) limited message length; ii) affordable computational cost; and iii) limited battery consumption.

As it has been explained previously, the orchestration protocol is run once for a whole service request, while the monitoring protocol runs in a continuous way to deal with uninterrupted streams of data. Due to this fact, the total workload of the proposed system is divided by narrowing the expensive cryptographic operations to the orchestration process, while leaving the monitoring process as an efficient and scalable protocol that only uses lightweight operations.

The *Secure orchestration* process requires the use of expensive ABE operations and its scalability may be affected by the number of attributes and by the number of simultaneous users in the system. This issue is shared with all the works in the literature that use ABE in lightweight devices such as [46, 35, 47, 19]. In particular, a similar orchestration protocol as the one presented in this paper was proposed in [19]. There, the authors studied the scalability of their orchestration process according to message length and computational cost, and they concluded that, despite the significant cost of the orchestration step, the limited number of runs expected for this procedure makes it still affordable for lightweight devices.

Regarding the *Secure monitoring* protocol, this procedure has a very relevant incidence on the scalability of the proposed system due to the fact that it has to be run intensively in monitoring services. For this reason, the rest of this section focuses on this protocol. In particular, its scalability and energy consumption are next studied. Also, a comparison between our proposal and two representative systems from the literature are provided.

6.1. Scalability

The scalability of the monitoring protocol is affected by two main aspects directly related to the level of effort required at sensor nodes in order to run the required computations and deal with the message length during the continuous transmission of sensed data. These two aspects are:

- nU: Number of users who concurrently use the system.
- $nFN + nSN$: Number of fog and sensor nodes present in the network.

The monitoring protocol has a *first step for multicasting* a single piece of data from U to the sensor nodes; and a *second and iterative step for transmitting data* back from the sensor nodes towards U in a continuous way.

The *multicast step* used in the monitoring protocol involves U sending a unique message $\{v, t_U, t_{rt}, HMAC(v||t_U||t_{rt}, K^v_{FN_{ep}})\}$ with fixed length to the sensor nodes by means of the entry point FN_{ep}. This message contains identifier v (e.g., integer of 32 bits), two time-related values (e.g., two integers of 32 bits each) and one HMAC value (e.g., 256 bits if using SHA256). This message is digitally signed by FN_{ep}. (e.g., if using DSA, the digital signature will have a message length of 1024 bits). Intermediate fog nodes and sensor nodes only verify that signature, store some data and forward the received message to other nodes in the hierarchy. The total length of the messages sent during this step is constant $O(1)$, it does not depend on $nFN + nSN$; and, more specifically, it is assumed to be around 172 bytes. On the other hand, the number of messages generated in this step directly depends on the number of users using the system simultaneously, due to the fact that each different U will perform her own and independent multicast step. As a result, the total message length that a sensor node will have to process in this step grows linearly according to $O(nU)$, this is $nU * 172$ bytes.

Regarding computation, the most costly operation in this step is the digital signature generation, which is only done by FN_{ep} (a capable device); other nodes just verify one digital signature per each multicast step being performed. According to that, the number of users nU using the system simultaneously generates a total computational cost at the sensor nodes of $O(nU)$ (each user performs her own and independent multicast step). Aspect $nFN + nSN$ has no effect here.

In summary, the costs at the sensor nodes in the *multicast step* are a message length of $nU * 172$ bytes, and nU digital signature verifications. Both costs are linear but, due to the fact that this step is ideally run only once for each monitoring service, and that those costs are quite reduced, we consider this step to be affordable for lightweight devices. Moreover, it is worth mentioning that a large network would be able fit a significant number of users simultaneously using different sets of nodes; and, if the network is small, the cloud may limit the number of simultaneous users in the system at the user joining step.

In the *iterative step for transmitting data*, at each iteration, sensor nodes generate a message $\{ID_{SN_i}, (d_1, \ldots, d_t), t_{SN_i}, \sigma_U, \sigma_{FN_a}, \ldots, \sigma_{FN_z}\}$ with fixed length and send it towards U. This message contains an identifier (e.g., integer of 32 bits), the encrypted sensor reading (i.e., t bits), the sensing time (e.g., integer of 32 bits), and a set of HMACs (e.g., with a length of 256 bits each) as large as the number l of ancestors that the specific sensor node has in the orchestrated hierarchy (i.e., fog nodes plus U). This represent a total message length of $64 + t + (l * 256)$ bits for each sensor reading transmission. Variable t can be assumed to have a fixed length of 32 bits. Variable l depends on the specific topology that has been used to deploy the network, which we consider to be a design decision; in any case, we assume that the number of fog nodes between a sensor node and U will be quite limited (in our experiments, we assume $l = 4$, this is, 3 fog nodes plus the user). As a result, we consider that t and l do not directly depend on $nFN + nSN$, which implies that the length of the messages sent during this step is constant $O(1)$, and the assumed total

length of each transmission is around 140 bytes. Nevertheless, similarly to the former step, the total number of transmissions that will be performed during this step depends directly on nU, due to the fact that, each user will run her own service monitoring independently of the others. As a result, the total message length that a sensor node has to process in this step grows linearly according to $O(nU)$, this is $nU * 140 * nT$ bytes, where nT stands for the average number of transmissions that each monitoring service will require.

Regarding computation, in the iterative step, sensor nodes are only required to compute low-cost bitwise operations to encrypt the data and generate low-cost HMACs values. The number of computations that each sensor node must perform depends on variables t, l and nU. The effect of those variables has been already discussed when studying the message length cost and the same conclusions applies here: the computational cost at sensor nodes grows linearly according to $O(nU)$, in particular, this step would require a sensor node to compute $nU * l * nT$ HMAC operations, being nU the relevant value here. It is worth mentioning that, even though the number of HMAC operations may result to be large, they are low-cost operations totally suitable for lightweight devices.

In summary, the costs required at the sensor nodes in the *iterative step for transmitting data* are a message length of $nU * 140 * nT$ bytes, and $nU * l * nT$ HMAC operations, being nU the relevant value in both cases. Those costs are linear with respect to nU, however, the number of bytes of each transmission is quite small, and only low-cost operations are involved. As a result, we argue that this step is affordable for lightweight devices. Moreover, similarly to the *multicast step*, in the case that a large nU becomes a scalability problem, the cloud may limit the number of simultaneous users in the system at the user joining step.

6.2. Energy consumption

As it has been discussed, the main operations to be considered during the monitoring protocol are: DSA/ECDSA signature generation, DSA/ECDSA signature verification and HMAC generation. Among those, DSA/ECDSA signature generation is only performed by the fog node acting as entry-point to the network, a node which is considered to be fully-fledged, i.e., it has no issues related to energy/battery consumption. On the other hand, DSA/ECDSA signature verification and HMAC generation operations are performed by lightweight sensor nodes in which energy consumption may be an issue. For this reason, in the following, we focus this study on the operations run by sensor nodes.

We have tested those operations in a *Raspberry Pi 1 Model B* device. The specifications of this device are: Broadcom BCM2835 processor, 700MHz, single-core, level 1 cache of 16 KB, level 2 cache of 128 KB, and 512MB RAM. The Raspberry used is equipped with *Raspbian*[6] as operating system. Operations

[6]https://www.raspberrypi.org/downloads/raspbian/

Table 2: Energy consumption and run-time for a single run of certain cryptographic operations tested in a *Raspberry Pi 1 Model B*

Operations	Run-time	Energy consumption
DSA signature verif.	0.06997s	9μAh
ECDSA signature verif.	0.01569s	1.6μAh
HMAC generation	0.00017s	0.02μAh
CP-ABE decrypt.-4 attr.	0.55172s	60μAh
CP-ABE decrypt.-10 attr.	1.27387s	130μAh
AES encryption	0.00052s	0.03μAh
CP-ABE encrypt.-4 attr.	0.60488s	65μAh
CP-ABE encrypt.-10 attr.	1.38582s	139μAh

related to computing HMACs, and generating/verifying ECDSA and DSA signatures have been implemented using *Python*[7] as programming language and the cryptographic libraries: *charm*[8] and *pycrypto*[9].

Table 2 summarizes the figures for energy consumption and run-time for DSA/ECDSA signature verification and HMAC generation. It also shows the obtained numbers for CP-ABE decryption operations with 4 and 10 attributes, just to highlight the significant difference in cost between the operations that a sensor node run (once) during the orchestration protocol and during the (continuous) monitoring protocol. The provided figures are the average between 100 and 500,000 runs depending on the cost of each operation (i.e., 100 runs for the CP-ABE operations, and 500,000 runs for the HMAC generation). The figures show that the operations run by sensor nodes during the monitoring protocols are lightweight, both in run-time and energy consumption.

In order to put those results into perspective, let us assume a simple orchestrated tree topology in which the sensor node is connected to U through 3 intermediate fog nodes. In this case, the sensor node would build 4 HMACs for each data preparation; and an additional ECDSA signature verification would be required each time that U re-starts the monitoring service. The worst case would imply that U re-starts the monitoring service for each iterative data transmission (note that, in this case, this cannot be considered a continuous monitoring). This worst case would require building 4 HMACs and verifying 1 ECDSA signature with an approximate total consumption of 1.68μAh for each data preparation calculation. Assuming a 5V battery of 10,000mAh, this worst case would allow a sensor node to serve more than 5 million data preparation calculations (ignoring idle waiting, sensing and data transmission consumption). Regarding the best case, this is, U starts the monitoring service only once and the sensor node continuously sends its readings without interruption, this case would cost 1.68μAh for the first data preparation (i.e., building 4 HMACs plus verifying 1 ECDSA), but only 0.08μAh for all the subsequent data transmis-

[7] https://www.python.org/
[8] https://github.com/JHUISI/charm
[9] https://github.com/dlitz/pycrypto

Table 3: Comparison with other proposals according to the considered testbed

Proposal	Energy per preparation	No. of preparations
New scheme (worst case)	$\approx 1.68\mu$Ah	≈ 5.9 million
New scheme (best case)	$\approx 0.08\mu$Ah	≈ 125 million
[19]: orchestr. + request-response	$\approx 3.22\mu$Ah	≈ 3.1 million
[35]: no orchestr. + CP-ABE-4attr.	$\approx 65\mu$Ah	≈ 153846
[35]: no orchestr. + CP-ABE-10attr.	$\approx 139\mu$Ah	≈ 71942

sions (i.e., building 4 HMACs). This would allow a sensor node to serve around 125 million data preparation calculations. Those numbers show that the new proposal is lightweight enough to be deployed in IoT environments in which continuous streams of data are generated and processed.

6.3. Comparison with other proposals

In order to reflect the level of improvement achieved by the new proposal, in this section we compare it against two representative schemes from the literature: the method presented in [19], which applies fog orchestration as well but focuses only on single request-response services; and the approach presented in [35] in which medical sensors collect measurements, encrypt them with CP-ABE, and send them to a data collection server.

In the scheme presented in [19] there is a first step in which the network is orchestrated in a similar way to our proposal. Next, the orchestrated sensor nodes send their readings towards the user following a pure request-response model. Applying this approach to perform continuous monitoring of data would require the user to send as many challenges to the sensor nodes as sensor readings she is expecting, and the corresponding acknowledgments indicating that the data has been properly received. As a result, in this scheme, in order to run a single data preparation, sensor nodes are required to compute 2 DSA/ECDSA signature verifications and 1 HMAC. According to the tests performed in our testbed, each data preparation consumes around 3.22μAh. Comparing this energy consumption with the figures obtained by the new proposal in the worst case, 1.68μAh, and in the best case, 0.08μAh, shows that the new proposal represents a clear improvement with respect to [19] regarding energy usage at lightweight devices when running monitoring services.

Regarding the proposal presented in [35], in this case there is no previous orchestration of the network, which implies that the costly CP-ABE cryptography is used at the sensor nodes for each data preparation. In particular, for each data transmission, sensor nodes encrypt the data to be sent with AES[10], encrypt the AES key with CP-ABE, and send the pair to a data collection server. According to the tests performed in our testbed (see Table2 for costs related to those operations), computing an AES encryption in a sensor node consumes around 0.03μAh, which is practically negligible, hence, the representative value

[10] http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

corresponds to the cost of computing the CP-ABE encryption required at each data transmission. In this way, computing a CP-ABE encryption with 4 attributes consumes around 65μAh, which implies that a data preparation in [35] is around 38 times costlier than in our proposal in the worst case (i.e., 1.68μAh); this already important difference grows greatly if we consider the best case (i.e., 0.08μAh per data transmission). Moreover, if a CP-ABE encryption with 10 attributes is required, then the energy consumption of [35] per data preparation grows up to 139μAh. This high energy consumption would deplete the 5V battery of 10,000mAh in approximately 72 thousand data transmissions. Comparing this result with the 125 million data preparation calculations that our proposal may perform in the best case shows the importance of introducing a preliminary orchestration step before running the service monitoring process.

Table 3 summarizes the comparison performed between the three different approaches in the considered testbed (i.e., *Raspberry Pi 1 Model B* plus 5V battery of 10,000mAh). This table depicts the energy required by each proposal to perform a single data preparation, and the total number of data preparations that each approach may run until battery depletion. For the new proposal, we have taken into account the best and worst situations. Regarding [35], we have considered the use of a CP-ABE encryption with 4 or 10 attributes.

7. Concluding remarks and future work

The protocols we propose make it possible to deploy fog-based IoT monitoring services in untrusted environments. To conciliate security and low latency, we leverage the flexibility and scalability of ABE to orchestrate the network that will perform the service, and the efficiency of symmetric cryptography employed during the monitoring process. The latter introduce very small overhead to the system, which is crucial when sensors are low tier and energy constrained IoT devices. On the other hand, fog orchestration avoids the need to broadcast requests and data updates through the whole network.

As future lines of research, we plan to adapt the orchestration protocol to other non-strictly hierarchical fog architectures considered in monitoring services (e.g., where fog nodes at the same level can communicate among each other) [6] and consider other fog-related architectures, such as mist computing [48]. We also plan to explore solutions to support mobile IoT devices (such as wearables that can move from the area controlled by one fog node to another one), while maintaining the consistency of the hierarchical organization of the network; and to improve the scalability of the proposed system with regard to the number of users who use the system concurrently, by means of aggregating user requests related to similar services.

Acknowledgments and disclaimer

This work was partly supported by the European Commission (project H2020-871042 "SoBigData++"), the Spanish Government (projects RTI2018-095094-

B-C21 “CONSENT” and TIN2016-80250-R “Sec-MCloud”) and the Government of Catalonia (2017 SGR 705). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of UNESCO. Some of the icons used in the figures of this paper are made by Freepik, Yannick, SimpleIcon, Recep Kutuk and Daniel Bruce from www.flaticon.com and licensed by CC 3.0 BY.

References

[1] A. Mutlag, M. Ghani, N. Arunkumar, M. Mohammed, O. Mohd, Enabling technologies for fog computing in healthcare iot systems, Future Generation Computer Systems 90 (2019) 62–78.

[2] F. Kraemer, A. Braten, N. Tamkittikhun, P. D., Fog computing in healthcare - a review and discussion, IEEE Access 99 (2017) 9206–9222.

[3] G. López, V. Custodio, J. Moreno, Lobin: E-textile and wireless sensor network-based platform for healthcare monitoring in future hospital environments, IEEE Transactions on Information Technology in Biomedicine 14 (6) (2016) 1466–1458.

[4] A. Rahmani, T. Gia, N. B., A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach, Future Generation Computer Systems 78 (2018) 641–658.

[5] A. Monteiro, H. Dubey, L. Mahler, Q. Yang, K. Mankodiya, Fit: A fog computing device for speech tele-treatments, in: Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP), 2016, pp. 1–3.

[6] P. Verma, S. Sood, Fog assisted-io-t enabled patient health monitoring in smart homes, IEEE Internet of Things Journal 5 (3) (2018) 1789–1796.

[7] R. Basir, S. Qaisar, M. Ali, M. Aldwairi, M. Ashraf, A. Mahmood, M. Gidlund, Fog computing enabling industrial internet of things: State-of-the-art and research challenges, Sensors 19 (2019) 4807.

[8] R. Alonso, I. Sittón-Candanedo, O. García, J. Prieto, S. Rodríguez-González, An intelligent edge-iot platform for monitoring livestock and crops in a dairy farming scenario, Ad Hoc Networks 98 (2020) 102047.

[9] I. Stojmenovic, S. Wen, The fog computing paradigm: Scenarios and security issues, in: Proceedings of the 2014 IEEE Federated Conference on Computer Science and Information Systems, 2014, pp. 1–8.

[10] H. Al Hamid, S. Rahman, M. Hossain, A. Almogren, 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 (2017) 22313–22328.

[11] A. Elmisery, S. Rho, D. Botvich, A fog based middleware for automated compliance with oecd privacy principles in internet of healthcare things, IEEE Access (2016) 8418–8441.

[12] A. Elmisery, S. Rho, M. Aborizka, A new computing environment for collective privacy protection from constrained healthcare devices to iot cloud services, Cloud Computing (2017) 1–28.

[13] T. Gia, I. Dhaou, M. Ali, A. Rahmani, T. Westerlund, P. Liljeberg, H. Tenhunen, Energy efficient fog-assisted iot system for monitoring diabetic patients with cardiovascular disease, Future Generation Computer Systems 93 (2019) 198–211.

[14] H. Tao, M. Bhuiyan, A. Abdalla, M. Hassan, J. Zain, T. Hayajneh, Secured data collection with hardware-based ciphers for iot-based healthcare, IEEE Internet of Things Journal (2018) 410–420.

[15] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, M. Rovatsos, Fog orchestration for internet of things services, IEEE Internet Computing 21 (2) (2017) 16–24.

[16] A. Alrawais, A. Alhothaily, C. Hu, X. Cheng, Fog computing for the internet of things: Security and privacy issues, IEEE Internet Computing 21 (2) (2017) 34 – 42.

[17] K. Fan, J. Wang, X. Wang, H. Li, Y. Yang, A secure and verifiable outsourced access control scheme in fog-cloud computing, Sensors 17 (7) (2017) 1695.

[18] H. Wanga, Z. Wanga, J. Domingo-Ferrer, Anonymous and secure aggregation scheme in fog-based public cloud computing, Future Generation Computer Systems 78 (2) (2018) 712–719.

[19] A. Viejo, D. Sánchez, Secure and privacy-preserving orchestration and delivery of fog-enabled iot services, Ad Hoc Networks 82 (2019) 113 – 125.

[20] X. Liu, A. Dastjerdi, R. Buyya, Chapter 8 - stream processing in iot: foundations, state-of-the-art, and future directions, in: R. Buyya, A. V. Dastjerdi (Eds.), Internet of Things, Morgan Kaufmann, 2016, pp. 145–161.

[21] R. Nehme, H. Lim, E. Bertino, Fence: Continuous access control enforcement in dynamic data stream environments, in: Third ACM conference on Data and application security and privacy, 2013, pp. 243–254.

[22] J. Cao, B. Carminati, E. Ferrari, K. Tan, Acstream: Enforcing access control over data streams, in: IEEE International Conference on Data Engineering, 2009, pp. 1495–1498.

[23] B. Carminati, E. Ferrari, J. Cao, K. Tan, A framework to enforce access control over data streams, ACM Transactions on Information and System Security 13 (3) (2010) Article No. 28.

[24] W. Ng, H. Wu, W. Wu, S. Xiang, K. Tan, Privacy preservation in streaming data collection, in: 2012 IEEE 18th International Conference on Parallel and Distributed Systems, 2012, pp. 810–815.

[25] V. Kalogeraki, D. Gunopulos, R. Sandhu, B. Thuraisingham, Qos aware dependable distributed stream processing, in: 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing, 2008, pp. 69–75.

[26] X. Xie, I. Ray, R. Adaikkalavan, R. Gamble, Information flow control for stream processing in clouds, in: 18th ACM symposium on Access control models and technologies, 2013, pp. 89–100.

[27] M. Ahmad, M. Amin, S. Hussain, B. Kang, T. Cheong, S. Lee, Health fog: a novel framework for health and wellness applications, Journal of Supercomputing 72 (10) (2016) 3677–3695.

[28] A. Shai, B. Waters, Fuzzy identity-based encryption, in: Advances in Cryptology-EUROCRYPT, Springer, 2005, pp. 457–473.

[29] D. Anh, A. Datta, Streamforce: outsourcing access control enforcement for stream data to the clouds, in: 4th ACM conference on Data and application security and privacy, 2014, pp. 13–24.

[30] C. Thoma, A. Lee, A. Labrinidis, Cryptstream: Cryptographic access controls for streaming data, in: 5th ACM Conference on Data and Application Security and Privacy, 2015, pp. 163–165.

[31] C. Thoma, A. Lee, A. Labrinidis, Polystream: Cryptographically enforced access controls for outsourced data stream processing, in: 21st ACM on Symposium on Access Control Models and Technologies, 2016, pp. 227–238.

[32] Y. Yang, H. Zhu, H. Lu, J. When, Y. Zhang, K.-K. Choo, Cloud based data sharing with fine-grained proxy re-encryption, Pervasive Mobile Computing 28 (2016) 122–134.

[33] Y. Jiang, W. Susilo, Y. Mu, G. Guo, Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing, Future Generation Computer Systems 78 (2018) 720–729.

[34] C. Zuo, J. Shao, G. Wei, M. Xie, M. Ji, Cca-secure abe with outsourced decryption for fog computing, Future Generation Computer Systems 78 (2018) 730–738.

[35] M. Ambrosin, A. Anzanpour, M. Conti, On the feasibility of attribute-based encryption on internet of things devices, IEEE Micro 36 (6) (2016) 25 – 35.

[36] X. Wang, J. Zhang, E. M. Schooler, M. Ion, Performance evaluation of attribute-based encryption: Toward data privacy in the iot, in: 2014 IEEE International Conference on Communications, IEEE Computer Society, 2014, pp. 725–730.

[37] L. Touati, Y. Challal, A. Bouabdallah, C-cp-abe: Cooperative ciphertext policy attribute-based encryption for the internet of things, in: 2014 International Conference onAdvanced Networking Distributed Systems and Applications (INDS), 2014, pp. 64–69.

[38] H. Shafagh, A. Hithnawi, L. Burkhalter, P. Fischli, S. Duquennoy, Secure sharing of partially homomorphic encrypted iot data, in: Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, 2017, p. Article No. 29.

[39] A. Paul, H. Pinjari, W.-H. Hong, H. Seo, S. Rho, Fog computing-based iot for health monitoring system, Journal fo Sensors (2018) 1–7.

[40] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy, 2007, pp. 321–334.

[41] R. Blom, An optimal class of symmetric key generation systems, in: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), 1984, pp. 335–338.

[42] A. Viejo, F. Sebé, J. Domingo-Ferrer, Secure and scalable many-to-one symbol transmission for sensor networks, Computer Communications 31 (2008) 2408–2413.

[43] A. Boldyreva, V. Kumark, A new pseudorandom generator from collision-resistant hash functions, in: Proceedings of the Cryptographers' Track of the RSA Conference (CT-RSA '12), 2012, pp. 187–202.

[44] J. Katz, Y. Lindell, Introduction to Modern Cryptography, Chapman & Hall/CRC, 2014.

[45] M. Yang, A. Anbuky, W. Liu, Security of the multiple-key blom's key agreement scheme for sensor networks, in: Proceedings of the IFIP International Information Security Conference (SEC '14), 2014, pp. 66–79.

[46] X. Yao, Z. Chen, Y. Tian, A lightweight attribute-based encryption scheme for the internet of things, Future Generation Computer Systems 49 (2015) 104–112.

[47] N. Oulha, K. T. Nguyen, Lightweight attribute-based encryption for the internet of things, in: 2016 25th International Conference on Computer Communication and Networks (ICCCN), 2016, pp. 1–6.

[48] A.-U. Rahman, F. Afsana, M. Madmud, M. Kaiser, M. Ahmed, O. Kaiwartya, A. James-Taylor, Towards a heterogenous mist, fog, and cloud based framework for the internet of healthcare things, IEEE Internet of Things Journal.