Secure and Privacy-Preserving Orchestration and Delivery of Fog-Enabled IoT Services

Alexandre Viejo, David Sánchez

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-enabled IoT applications are the next step in the deployment of cloud computing backed services through the Internet. The use of fog nodes at the edge of the network reduces traffic to cloud servers, decreases latency of services and improves ubiquity. Security is, however, a fundamental challenge of fog computing because both fog and IoT nodes may be deployed in disperse non-secure locations. Solutions to this challenge rely on expensive cryptography, such as attribute-based encryption of homomorphic encryption, which significantly degrades the response time of the service delivery due to the limited resources of IoT nodes. In this paper, we tackle this issue by relying on the novel concept of *fog orchestration*. Through orchestration, the network is self-tailored to the service to be delivered, and we use this possibility to enable a secure and efficient service delivery. Specifically, we propose several secure and privacy-by-design protocols for the orchestration and delivery fog-enabled IoT services. We also assume the most challenging scenario in which nodes exchange data in open and potential unsecured networks and they can be subjected to a wide range of active attacks. The feasibility of our proposal and the improvements it brings over related works are discussed through a set of theoretical and empirical evaluations of its performance.

Keywords: Security, Privacy, Fog Computing, Orchestration, Internet of Things.

1. Introduction

Cloud computing has revolutionized the delivery of online services through the Internet. Internet of Things (IoT) services, such as systems monitoring, surveillance, healthcare or smart cities-related services, are highly demanded and among the ones that benefit most from the availability, ubiquity, scalability and cheap storage and computational power offered by the Cloud. However, many IoT services are highly distributed and rely on a complex network of many low tier devices, such as sensors, for which a constant connection to cloud service providers is impractical. To mitigate this issue, fog computing extends cloud computing by moving a substantial amount of computation, communication and short period storage to the edge of the network, that is, to the so-called fog nodes. This allows reducing data volume and traffic to the cloud servers, decrease latency and response delay [1] and improve the ubiquity of the system, by which the users may access the services through different entry points (i.e., the cloud but also fog nodes).

Fog computing-based IoT applications consist of a cloud service provider, one or several fog nodes at the edge of the network connected to the cloud and a potentially complex network of IoT devices below them. Fog nodes are in charge of routing requests to IoT devices and collecting and aggregating data from such devices; these data are then transmitted to the cloud, which provisions resources (computing and storage) to store and analyze the collected data and enforces high-level services based on a predefined system logic [2].

Among the several challenges inherent to fog computing for IoT, privacy and security arise as fundamental issues [3]. Complex IoT services usually involve multiple components (sensors, communication devices, computer chips, etc.), which may be deployed in different non-secure geographic locations, thus increasing the probability of attacks. More specifically, authentication, access control, rogue node and intrusion detection, and revocation are identified as main security challenges [4], and they require a careful design due to the distributed nature of the fog computing architecture and the lack of computational and energy resources of the IoT devices.

Moreover, fog nodes and IoT devices, due to their limited hardware, disperse locations and WAN-based communication, are specially vulnerable to cyberattacks or, even, physical attacks. As a result, fog nodes may be untrusted, thus causing serious privacy threats when the users' requests and/or the sensed data (which are transmitted and broadcast through the network), contain sensitive information such as identities, locations, or healthcare data [2].

This work focuses on mitigating these issues by proposing secure and privacy-by-design protocols for the delivery fog-enabled IoT services in distributed and open networks.

1.1. Background and related work

Fog-enabled IoT services share similarities with the data stream processing scenario, in which stream producers serve data in real-time, and stream consumers gather these data [5].

In the past, data encryption and standard access control models, such as Role-based access control (RBAC), have been used to secure data stream processing [6, 7, 8, 9, 10, 11]. The main shortcoming of these works is that they rely on a fully-trusted stream processing infrastructure to enforce access control on data streams. Because this approach can only deal with external adversaries, it is not suitable for scenarios in which IoT devices may be compromised and may misbehave and perform security and privacy attacks.

The scientific community has also started working on proposals in which the stream processing infrastructure is not trusted anymore, and in which the access control enforcement relies on cryptography. More specifically, proposals such as [12, 13, 14] consider an untrusted, honest-but-curious data stream management system that uses Attribute-Based Encryption (ABE) [15] to enforce access control. ABE is a convenient solution because: i) it supports fine-grained access control based on the recipients' attributes without relying on a centralized authority; ii) it scales on the number of attributes considered in the system, rather than on the number of entities to manage (as in standard cryptography); and iii) it is robust against collusion attacks conducted by entities with complementary attributes that try to decrypt a ciphertext that, individually, are not allowed to.

In any case, the use of centralized data stream processing infrastructures does not fit well with the distributed and ubiquitous nature of fog-enabled IoT services, in which users may directly interact with IoT devices and fog nodes situated at different layers of the network. In these scenarios, in which there is not a centralized entry point to the system, security and access control should be self-enforced by the entities participating in the service delivery. For such purpose, the scientific community on fog computing has also adopted ABE as the foremost security-enabling solution in fog architectures [16, 17, 18]. However, the computational cost of ABE is the main barrier that hinders its practical deployment. In particular, the decryption cost of the ABE ciphertext is proportional to the number of attributes involved in the access policies; and this cost may hamper the

latency of fog services when decryption operations are performed on IoT devices with limited resources [19, 20].

To lower this cost, some authors have proposed outsourcing the decryption operation to a powerful proxy (such as the cloud) in a secure way [21]. Even though this solution enabled IoT devices to be relieved from most of the decryption cost of ABE ciphertexts, it goes against the philosophy of fog computing, which is to minimize (or avoid) the communication of the dispersed IoT devices with the cloud.

Other cryptographic solutions, such as the use of fully homomorphic encryption [22], are significantly costlier than ABE and are still far from achieving reasonable performances if fog-enabled services [23].

On the other hand, a promising solution to minimize the latency of fog-based services is the use of *fog orchestration*. Fog orchestration arises as the key to align the devices' operation with the service and users' interests [2]. Specifically, fog orchestration assumes that most IoT services require only the collaboration of a reduced subset of the devices that form the whole network. For example, if a user is willing to gather the temperatures of certain rooms in a certain industrial facility, all the fog and IoT nodes situated outside the areas of interest should not be used. This behavior allows the devices to save their resources; moreover, working with a small number of IoT nodes will directly reduce the quantity of data to be re-transmitted by the fog nodes situated at the top layers of the fog computing architecture. Through orchestration, the network determines and selects the best IoT appliances to dynamically compose a simplified workflow for the delivery of the service requested by the user. However, practical research on *fog orchestration* for IoT is still at its infancy and most works published so far are on the conceptual side [2].

1.2. Contributions and plan of this paper

Fog orchestration has the potential to minimize security and privacy issues because, after orchestrating the network for a service, the user's requests are only forwarded to the subset of nodes in charge of fulfilling the request [2]. However, the orchestration process itself, which implies broadcasting the users' needs for service discovery, is still a critical task. We tackle this issue by leveraging the flexibility and scalability of ABE (that is used only during the orchestration process), with the efficiency and convenience of standard public and symmetric cryptography (which are employed once the orchestration of the service has been done). Being the ABE-related operations the most expensive ones, with this approach we minimize the latency and energy consumption of (low tier) IoT devices. This differentiates our

approach from other systems also employing ABE in fog-based IoT services that, due to the lack of service orchestration, use –the more expensive ABE– cryptography [4, 24] or even homomorphic encryption [22] during the whole life cycle of the service to enforce access control.

As main contributions, we propose several secure and privacy-by-design protocols for the orchestration and delivery fog-enabled IoT services. On the contrary to related works, we assume the most challenging scenario in which the entities in the system (the cloud, fog nodes and IoT devices) exchange data in open and potential unsecured networks and can be subjected to a wide range of passive and active attacks, which include compromising and controlling fog or IoT nodes. Moreover, we consider a fully ubiquitous scenario in which the users may request services not only to the cloud, but also to fog nodes in their vicinity. Finally, we consider both security and privacy issues, where the latter refer to the protection of the preferences of the users during the request and delivery of fog services. On the technical side, our proposal leverages ABE, symmetric and public cryptography to i) secure the (potentially sensitive) data exchanged and broadcast through the network; ii) detect and counter-measure attacks undergone by malicious parties; and iii) ensure that only the data that are strictly necessary to fulfill the service is revealed to the involved entities, thus fulfilling the *data minimization* principle included in the EU General Data Protection Regulation (GDPR)[1].

The rest of the paper is organized as follows. Section 2 describes the general architecture of fog-enabled systems. Section 3 details our approach to secure orchestration and delivery of fog-enabled services. Section 4 formalizes the protocols that constitute our solution. Section 5 is devoted to analyzing the security and privacy of the proposal. Section 6 analyzes the performance of our solution. Finally, Section 7 reports the conclusions of this work and proposes some lines of future research.

2. System architecture

Figure 1 details the architecture of the fog-enabled systems we consider in this paper. In the following we detail the entities involved in such systems.

- *Users.* They may use any personal device such as a laptop or a smartphone to contact the fog computing system and request the execution

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

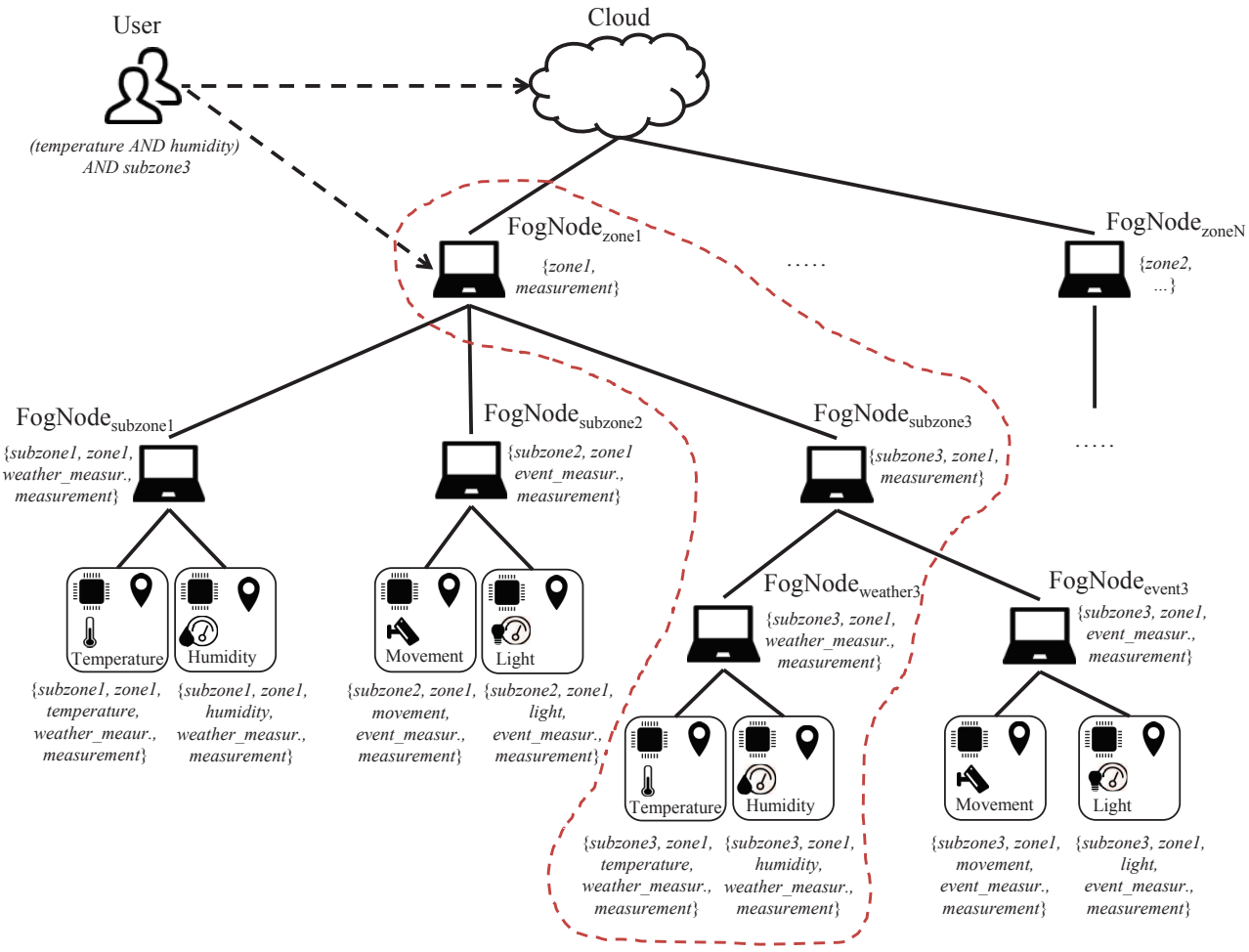


Figure 1: Sample architecture of a hierarchical fog-enabled system. Zones and subzones correspond to locations of the sensors. The remaining attributes detail their sensing capabilities.

of a certain service, such as monitoring the temperature of a certain area. In our proposal, users may use either the cloud or a fog node as the entry point to the system.

- *Cloud.* This is the remote data center hosted by the organization that is in charge of the whole fog computing system. It is assumed to be resilient against attackers and it has full computational capabilities. In our proposal, the cloud will be also in charge or generating and managing the cryptographic materials needed to enforce secure communications across the nodes.

- *Fog nodes.* These devices are located at the edge of the Internet and their purpose is to bridge the gap between the cloud and the IoT nodes that are deployed at the bottom of the hierarchy. Fog nodes are assumed to be fully-fledged devices (so that they can deal with highly computationally and/or energy demanding operations) and are hierarchically organized according to the sensing devices they control. For example, different fog nodes can be deployed to cover a certain sensing areas (e.g., petrochemical plants or relevant zones within such plants) and/or to control different sensing magnitudes (e.g., weather-related or event-related measurements). In our proposal, fog nodes are in charge of orchestrating users' requests by routing them to the appropriate nodes/devices below them in the hierarchy. Fog nodes may be deployed in insecure areas and they are expected to be connected through open and untrusted networks, such as the Internet; therefore, they may be subjected to both active and passive attacks and, in the worst case, they may be compromised and fully controlled by an adversary. In the best case, they are expected to behave in an honest-but-curious way, this is, they will follow the protocol but may try to gather users' data, such as service requests.

- *Sensor nodes.* They are IoT devices capable of sensing a single magnitude (e.g., temperature, humidity, movement, light, etc.) and are deployed in specific locations, which are known by the nodes. Being lightweight, they only answer to sensing requests received by their corresponding fog node and they are not suited for highly computationally and/or energy demanding operations. As well as fog nodes, sensor nodes may be deployed in insecure areas and they are expected to be connected through untrusted network, which may result in being the target of different kinds of attacks and even becoming fully controlled by capable adversaries.

3. Our proposal

As shown in Figure 1, in our proposal, the users willing to gather sensed data from the fog computing network may enter the system remotely by means of the cloud using an Internet connection; or they may directly connect with a fog node located in their immediate vicinity using a short-range wireless connection. Whatever the method used, the fog computing system will build, through orchestration, a network of devices that is specifically tailored to provide the data required by the service by the user. This specific-purpose network (tagged with a dotted line in Figure 1) will be made of a subset of fog nodes and sensor nodes, and it will be constructed by following the tree-like topology in which the leaves of the tree are the sensor nodes that sense the data and the intermediate nodes (i.e., fog nodes) are routers that forward the user's requests, collect the messages coming from their children and aggregate them into a single message that is transmitted up to their parent.

3.1. Secure orchestration

As introduced in Section 1.2, node orchestration requires broadcasting users' requests through the network, which may cause security and privacy threats. Moreover, both fog nodes and sensors act as data controllers and data processors, they may be honest-but-curious and can be subjected to attacks and become fully untrusted.

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [25] fits well with the security needs of secure fog orchestration in such an adverse scenario. The entities in our system (sensor and fog nodes) are given keys associated to the attributes that represent their abilities to process the users' requests. With regard to sensor nodes, attributes may correspond to the magnitudes they can sense and/or their locations; with regard to fog nodes, attributes would represent the areas and/or the type of sensors they control, that is, they generalize the abilities of the nodes below them within the hierarchy. For such purpose, attributes are hierarchically organized consistently with the topology of the network. For example, concrete locations ($subzone1$, $subzone2$), which may be assigned to specific sensors, as shown in Figure 1, are generalized into areas ($zone1$), which are associated to the fog node that encompasses them; likewise, the specific magnitudes measured by the sensors ($temperature$, $humidity$) are generalized into types ($weather_measurement$), which are associated to their parent fog nodes.

Consistently with the hierarchical organization of nodes and attributes, sensors 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 node capable of measuring *temperature* in *subzone*1, will be given a secret key linked to the attributes {*temperature*, *weather_measurement*, *measurement*, *subzone*1, *zone*1}, as shown in Figure 1.

The user's request is used to orchestrate the nodes in the network. During this process, the fog nodes and sensor nodes capable of managing the service request are determined, whereas the remaining nodes in the hierarchy are pruned. We enforce this by using a service identifier v, which will be used to generate the cryptographic materials that will be used during the service delivery; and CP-ABE to encrypt v, which allows us to ensure that only the nodes with the proper capabilities (i.e., attributes) will be able to decrypt v and, therefore, participate in the service. The latter brings security, but also privacy to the orchestration, because the user's interests are only disclosed to the appropriate nodes.

More in detail, the service identifier v is encrypted according to a policy stating the Boolean combination of attributes that receivers should meet, and which would indicate that they are able to manage the service. To properly route the request through the hierarchy of nodes, as many policies as combinations of attribute generalizations of the specific attributes in the service request are generated. For example, as shown in Figure 1, let us assume that the user's service request consists on sensing *temperature* and *humidity* in the *subzone3* of *zone1*. According to the attribute generalization depicted in the figure, the following policies encompassing all the combinations of attribute values and attribute generalizations will be defined:

- $P_1 = ((temperature\ OR\ humidity)\ AND\ subzone3)$
- $P_2 = (weather_measurement\ AND\ subzone3)$
- $P_3 = (measurement\ AND\ subzone3)$
- $P_4 = ((temperature\ OR\ humidity)\ AND\ zone1)$
- $P_5 = (weather_measurement\ AND\ zone1)$
- $P_6 = (measurement\ AND\ zone1)$

The service identifier v is then encrypted with the policies, which generates as many ciphertexts as policies. The outputted ciphertexts are then sent to the contact fog node in the hierarchy, which will try to decrypt them.

If at least one ciphertext is successfully decrypted, it means that the node has some descendants that are able to manage the service request. Then, the ciphertexts that remain to be decrypted, which correspond to policies expressing more specific capabilities, are recursively broadcast to all the children nodes through the hierarchy. If a node receiving the ciphertexts is not able to decrypt any of them, it does nothing, because it implies that the branch of nodes below cannot fulfill the service and they will not participate in the delivery of the service. Finally, if a sensor node is able to decrypt the ciphertexts it receives, it means that it is a sensor capable of handling the service. Next, selected sensor nodes securely answer to the requesting user providing their identifiers. As a result of this process, the user knows the identifiers of the subset of nodes that will participate in the service delivery. With this information, the user requests the cloud to generate the temporal symmetric keys of the participant sensor nodes bounded to the service identifier v, while the sensor nodes are able to generate this cryptographic material by their own. More details on the specifics of this protocol are given in Section 4.2.

With this procedure, the service delivery would be enforced on the participant nodes in a secure way by relying on standard cryptography (detailed in Section 3.2), whereas the nodes that were not able to decrypt the CP-ABE ciphertexts and, thus, did not learn the user's request, were effectively pruned during the orchestration. In this way, we limit who can learn the interests of the user to the nodes that are strictly necessary to either route or fulfill the request; and, also, we minimize the data are revealed to such nodes, because fog nodes will only learn generalized interests of the user and sensor nodes will only get individual measurement requests (rather than the whole user request). This is consistent with the *data minimization* principle stated in the GDPR, because only the data that is strictly necessary to fulfill the service is employed by the entities involved in the system.

Formally, CP-ABE consists of four algorithms that will be instantiated during the execution of the protocols (See Section 4):

- $\{PK, MK\} = Setup$. They key management party uses a security parameter k determining the size of the bilinear group to generate the public key PK (which is published) and master key MK (which is kept in secret).

- $SK = KeyGeneration(MK, S)$. The key management party uses its master key MK and the set of attributes S associated to the entity to generate the CP-ABE key SK for such entity.

- $CT = Encrypt(PK, M, P)$. The data owner encrypts the message M by using the published PK and an access policy P, which consists of a Boolean predicate over the universe of attributes. A ciphertext CT is produced such that only a receiving entity that possesses the set of attributes satisfying the access policy will be able to decrypt it.

- $M = Decrypt(PK, CT, SK)$. The entity receiving the ciphertext CT decrypts it by using the published PK and its private key SK, which corresponds to its set of attributes S. If S satisfies the access policy P, which is implicit in CT, the original message M will be obtained.

In our setting, the key management party is the cloud; the receiving entities are either the fog or sensor nodes in the network; their set of attributes S will determine, respectively, their ability to route or to fulfill a request; the data owner will be the user and the message M will be the user's request.

3.2. Secure service delivery

Once the subset of nodes is orchestrated, the user and the participant sensor nodes share the right temporal symmetric keys bounded to the service identifier v.

Next, the user uses the orchestrated service by querying the participant nodes with a tuple of two integers that contain the identifier v of the service and the identifier of the specific protocol execution (i.e., integer p). The participant nodes will answer with the requested information to each query sent by the user, and each query will use a unique combination of (v, p).

These are the simplified steps of the process of serving each user query:

1. User sends the pair $v||p$ to the fog node situated at the top of the subset of selected nodes. This message contains an HMAC to provide authenticity and integrity.
2. The fog node that receives the pair $v||p$ verifies the HMAC, digitally signs the pair $v||p$, and multicasts it to the others nodes of the subset, which are organized following a tree hierarchy.
3. Upon reception, nodes of the subset first verify the signature; then, if they are intermediate fog nodes, they forward the received message to their children in the tree hierarchy; if they are final sensor nodes, they just store the pair $v||p$.
4. The final sensor nodes generate the requested data by the user (e.g., if they are temperature sensors, they sense the environment and gather the current temperature); then, they encrypt the data by means of the symmetric secret keys linked to this specific service, a bitwise XOR

operation and a hash function; they also generate an HMAC value to provide authenticity and integrity; finally, the encrypted data are sent to the user through the hierarchy formed by the subset of nodes.

5. The intermediate fog nodes aggregate the received answers using a bitwise operation and send the result to the top of the hierarchy.
6. The fog node situated at the top of the hierarchy delivers the final aggregated value to the user that can verify the HMACs generated by the sensor nodes and decrypt the data using the symmetric secret keys linked to the service and a bitwise XOR operation.

It is important to note that the *secure orchestration* process is executed only once in order to configure the subset of nodes that will serve the requests of the user. After that, the *secure service delivery* is executed as many times as requested by the user. Whereas the secure orchestration uses ABE cryptography, which is computationally costly, the secure service delivery only uses low-cost operations (hash functions, bitwise operations and signature verifications). With this, we significantly reduce the energy consumption at IoT nodes and the latency of the service w.r.t. related works that use ABE during the whole life cycle of the service. Moreover, the use of *secure orchestration* allows the proposed solution to only involve those nodes that can serve the request. The remaining nodes are not used at all and, therefore, can save their resources.

4. Protocols

In this section, we formalize the different procedures and protocols that rule the proposed system giving enough detail to allow their implementation. These are: *Setup and user registration*, *Secure orchestration* and *Secure service delivery*.

4.1. Setup and user registration

The setup phase happens before physically deploying the different fog and sensor nodes, and it is conducted by the organization that supports the fog computing system (e.g., the cloud entity).

The user registration is a step followed by users who want to use the fog computing infrastructure.

The objective of this step in both cases is to provide the different actors of the system with the cryptographic material required to execute the different protocols. As a result, the cryptographic resources that each entity stores after this initial phase are:

- *Cloud.*
  - It generates and stores all the symmetric secret keys (e.g., AES[2]) shared with all the fog nodes and sensor nodes.
  - It Builds a CP-ABE cryptosystem (see Section 3.1) and owns the corresponding public key PK^{cp-abe} and master key MK^{cp-abe}. It generates and stores all the CP-ABE keys assigned to each node of the fog computing system.
  - It configures a SSL/TLS communication channel with the corresponding X.509 certificates[3] to allow secure remote connections.

- *Fog node (FN_i).*
  - It shares a symmetric secret key K_{FN_i} with the cloud.
  - It generates a public/private key pair (i.e., PK_{FN_i}/SK_{FN_i}) from a public-key cryptosystem with signing capabilities (e.g., DSA[4]). SK_{FN_i} is stored by the fog node, and PK_{FN_i} is accepted as valid by all nodes located in the subtree rooted by FN_i.
  - It receives from the cloud a CP-ABE secret key $SK_{FN_i}^{cp-abe}$ related to its capabilities. Note that fog nodes are linked to generalized attributes that encompass all the capabilities of the devices located in the sub-tree rooted by them (see Section 3.1).

- *Sensor node (SN_i).*
  - It shares a symmetric secret key K_{SN_i} with the cloud.
  - It knows the public keys PK_{FN_i} of the fog nodes that are situated in the path towards the cloud.
  - It receives from the cloud a CP-ABE secret key $SK_{SN_i}^{cp-abe}$ related to its capabilities. In this case, CP-ABE keys are linked to concrete attributes specifying their capabilities, and also to all the generalized attributes of such capabilities (see Section 3.1).

- *User.*

[2]http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
[3]https://tools.ietf.org/html/rfc4158
[4]http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

- She stores a X.509 certificate to authenticate herself to the cloud and establish SSL/TLS connections.
- She stores the universe of attributes that define the capabilities of the fog nodes and sensor nodes that form the fog computing system; this includes all generalizations and specializations of each attribute (e.g., see Figure 1).
- She stores the public key PK^{cp-abe} linked to the CP-ABE cryptosystem.

4.2. Protocol-1: Secure orchestration

The objective of this protocol is to perform the *Secure orchestration* presented in Section 3.1. This is, select the subset of fog and sensor nodes among the whole fog computing system that will be used to run a specific application requested by the user. The subset of nodes will follow a tree hierarchy and they will receive the cryptographic material to be used during the execution of the *Secure service delivery* (see Section 3.2).

The secure orchestration process, which is depicted from a general point of view in Figure 2, is launched by a user U who wants to use the fog computing system for executing a specific application. Due to the ubiquitous nature of this type of networks, we consider two entry points for the user: she can connect with the cloud; or she can connect with a nearby fog node. First, we explain how the user joins the system in each case. Next, once the user has joined the system, we explain how the secure orchestration is done.

4.2.1. User enters the system by means of the Cloud

A user U enters the fog computing system by directly connecting to the Cloud as follows:

1. U establishes a SSL/TLS secured communication with the cloud.
2. The cloud verifies that U is allowed to use the fog computing system.
3. U generates a fresh random value v (freshness can be guaranteed by adding a timestamp), which will identify the fog computing application 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 user application.
4. U sends the pair $\{v, GATR_U\}$ to the cloud.
5. The cloud uses $GATR_U$ to decide which fog node FN_i is best suited to handle the requested fog computing application. In this way, the selected FN_b will act as *bootstrap node* between U and the fog computing system.

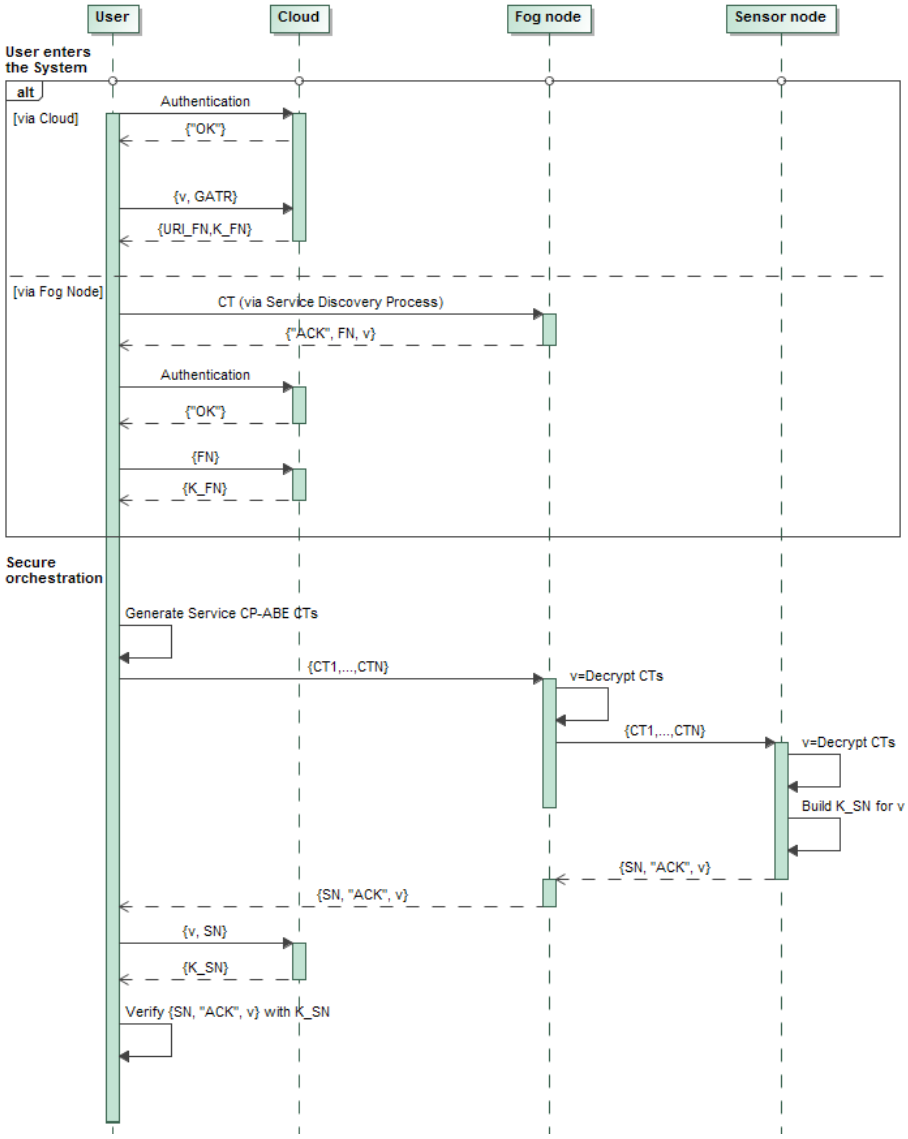


Figure 2: Overview of the interactions and messages sent among the entities in the system for the secure orchestration protocol.

6. The cloud checks that v is a fresh value (it has not been used previously) and builds the symmetric key $K^v_{FN_b} = H(v||K_{FN_b})$ that will be shared by FN_b and U to perform the requested application, where $||$ is the concatenation operator.
7. The cloud sends the pair $\{URI_{FN_b}, K^v_{FN_b}\}$ to U. Where URI_{FN_b} is a uniform resource identifier that may allow U to contact FN_b remotely.

4.2.2. User enters the system by means of a nearby fog node

A user U enters the fog computing system by connecting to a fog node located in her vicinity as follows:

1. U generates a fresh random value v (freshness can be guaranteed by adding a timestamp), which will identify the fog computing application 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 user application.
2. U builds a policy P_0 that suits with $GATR_U$ and generates the corresponding CP-ABE ciphertext CT_0 using PK^{cp-abe}. This is $CT_0 = Encrypt(PK^{cp-abe}, v, P_0)$.
3. U performs a *service discovery process*[5] that uses the CP-ABE ciphertext CT_0 in order to find a fog node FN_i in her vicinity capable of acting as *bootstrap node* between U and the fog computing system. More concretely, the service discovery process will detect those fog nodes close to U that are capable of decrypting CT_0 using their CP-ABE secret keys.
4. U selects one of the discovered fog nodes (i.e., FN_b). FN_b has already got the value v from the CP-ABE ciphertext CT_0 and will act as *bootstrap node* of U.
5. FN_b checks that v is a fresh value (it has not been used previously) and generates a temporal symmetric secret key $K^v_{FN_b} = H(v||K_{FN_b})$ bounded to v.
6. FN_b builds a message that contains an acknowledgment and its identification in the fog computing system; and uses key $K^v_{FN_b}$ to generate a HMAC[6] signature σ of this message. This is $\{\text{“}Ack\text{”}||FN_b||v, \sigma\}$, where $\sigma = HMAC(\text{“}Ack\text{”}||FN_b||v, K^v_{FN_b})$. This message is sent back to U.

[5]Service discovery procedures are usual in mobile ad-hoc networks [26]; it is out of the scope of this paper to define how this procedure would work.

[6]http://www.ietf.org/rfc/rfc2104.txt

7. U receives the message sent by FN_b, contacts the cloud by means of an Internet connection secured with SSL/TLS, sends the value v, and requests the symmetric key $K^v_{FN_b}$ that will be shared with FN_b.
8. The cloud first checks whether U is allowed to use the fog computing system. Next, it builds the required shared key $K^v_{FN_b} = H(v||K_{FN_b})$ and sends it back to U.
9. U uses $K^v_{FN_b}$ to verify the HMAC sent by FN_b and authenticates this node as a valid fog node.

4.2.3. User orchestrates the hierarchy of nodes

A user U, who has already joined the system, orchestrates a subset of nodes of the fog computing network to run a certain service v as follows:

1. 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.
2. U generates as many CP-ABE ciphertexts $\{CT_1, \ldots, CT_N\}$ as policies by computing:

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

3. U sends $\{CT_1, \ldots, CT_N\}$ to the bootstrap node FN_b together with an HMAC.
4. FN_b verifies the HMAC and tries to decrypt $\{CT_1, \ldots, CT_N\}$ using its $SK^{cp-abe}_{FN_b}$, which is associated to the (generalized) attributes that encompass the capabilities of the nodes below FN_b in the hierarchy; that is $Decrypt(PK^{cp-abe}, CT_x, SK^{cp-abe}_{FN_b})$.
    - If at least one of the CP-ABE ciphertexts is successfully decrypted, FN_b uses its private key SK_{FN_b} to digitally sign the non-decrypted CP-ABE ciphertexts, and multicasts them to its descendants in the hierarchy.
    - Else, FN_b does not continue the protocol.
5. Intermediate fog nodes FN_i receive the CP-ABE ciphertexts; they first verify the digital signatures; next, they try to decrypt them with their own 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. This process is repeated until the leaves of the hierarchy (i.e., the sensor nodes) are reached.

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

6. Sensor nodes SN_i receive the CP-ABE ciphertexts; they first verify the digital signatures; next, they try to decrypt them with their own CP-ABE secret keys.
   - If at least one of the CP-ABE ciphertexts is successfully decrypted, they will participate in the fog computing application using their sensors and transmitting their readings as requested. In this way:
     (a) SN_i builds a temporal symmetric secret key bounded to service v: $K^v_{SN_i} = H(v||K_{SN_i})$.
     (b) SN_i builds an acknowledgment containing its identifier (e.g., its serial number) and protected by means of an HMAC built with $K^v_{SN_i}$:

$$\{SN_i||\text{“}Ack''||v, HMAC(SN_i||\text{“}Ack''||v, K^v_{SN_i})\}$$

     (c) SN_i sends the acknowledgment towards U by means of its ancestor in the hierarchy. The message will be forwarded up to the top of the tree until FN_b is reached.
   - Else, SN_i does not continue the protocol and it is pruned by the orchestration process.
7. U receives the identifiers of all the sensor nodes that will participate in the fog computing application. Then, U sends to the cloud those identifiers and value v.
8. The cloud uses the sensor nodes' identifiers to retrieve their symmetric secret keys and, by means of v, it computes their temporal secret keys $K^v_{SN_i} = H(v||K_{SN_i})$. Then, the cloud sends them to U.
9. U uses keys $K^v_{SN_i}$ to verify the HMACs send by the sensor nodes. If the verification succeeds, U will use these temporal secret keys to perform the *secure service delivery* (see Sections 3.2 and 4.3).

4.3. Protocol-2: Secure service delivery

The objective of this protocol is to perform the *Secure service delivery* (see Section 3.2) by means of the subset of nodes that have been orchestrated in the past protocol. The type of services that we envisage in this paper are assumed to retrieve sensed data from the sensor nodes in the form of symbols codified as a binary sequence.

This protocol, which is depicted from a general point of view in Figure 3, is made of four main steps: i) user U challenges the orchestrated hierarchy; ii) sensor nodes send the sensed data towards U; iii) intermediate nodes aggregate the data in transit; and iv) U obtains the sensed data. These steps are next detailed.

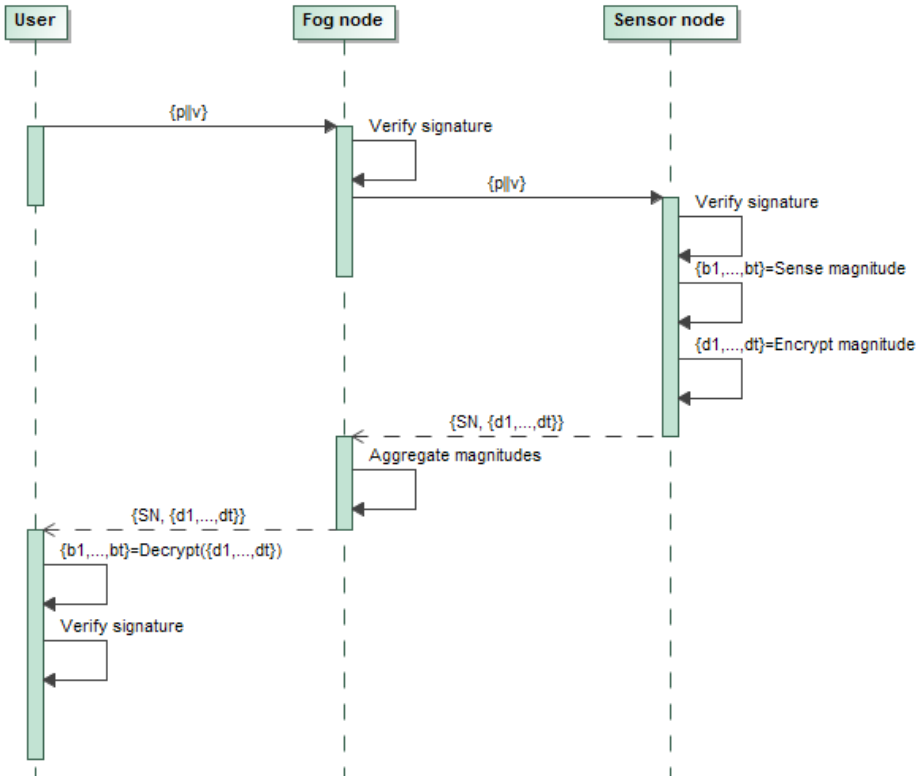


Figure 3: Overview of the interactions and messages sent among the entities in the system for the secure service delivery protocol.

4.3.1. User challenges the orchestrated hierarchy

User U challenges the orchestrated subset to retrieve the sensed data by following these steps:

1. U generates a random value p that identifies one specific protocol execution among the service identified by v.

2. U generates an HMAC of p using the secret key $K^v_{FN_b}$ shared with the bootstrap fog node FN_b. Then, it sends the pair $(p||v, HMAC(p||v, K^v_{FN_b}))$ to FN_b.
3. FN_b verifies the freshness of $p||v$ and the HMAC.
   - If both verifications succeed, FN_b signs $p||v$ using its secret key SK_{FN_b}, and multicasts the resulting $\{p||v\}_{SK_{FN_b}}$ to its descendants in the hierarchy.
   - Else, the message is discarded and FN_b does not continue the protocol.
4. A fog node FN_i that receives $\{p||v\}_{SK_{FN_b}}$ verifies its signature.
   - If the verification succeeds, FN_i 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.
   - Else, the message is discarded and FN_i does not continue the protocol.
5. A sensor node SN_i that receives $\{p||v\}_{SK_{FN_b}}$ verifies its signature.
   - If the verification succeeds, SN_i stores values p and v.
   - Else, the message is discarded and SN_i does not continue the protocol.

4.3.2. Sensor nodes send the sensed data towards the user

Let $(b_1, \ldots, b_t)$ be the binary representation of the symbol to be transmitted by a sensor node SN_i. Once SN_i receives the challenge $p||v$, it sends $(b_1, \ldots, b_t)$ to U by following these steps:

1. SN_i computes a pseudo-random t-bit sequence:

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

   where $lsb_t(\cdot)$ is a function returning the t least significant bits of its argument, H is a one-way hash function, and $K^v_{SN_i}$ is the temporal secret key shared between SN_i and U.
2. SN_i encrypts $(b_1, \ldots, b_t)$ in a sequence $(d_1, \ldots, d_t)$ as follows:

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

3. SN_i generates a HMAC of $(d_1, \ldots, d_t)$ using $K^v_{SN_i}$.

$$\sigma_i \leftarrow HMAC(d_1, \ldots, d_t||p||v||K^v_{SN_i})$$

4. SN_i sends the tuple $(SN_i; d_1, \ldots, d_t; \sigma_i)$ up to its ancestor in the orchestrated hierarchy.

4.3.3. Intermediate nodes aggregate the data in transit

Fog nodes FN_i receive the data originally sent by the sensor nodes and forward it towards U by following these steps:

1. FN_i receives messages from its descendants in the orchestrated hierarchy.
2. Once all the expected tuples $\{(SN_j; d_{1,j}, \ldots, d_{t,j}; \sigma_j)\}_j$ have been received, FN_i aggregates them by:
   (a) Concatenating the identifiers and symbols sent by each child.
   (b) Computing $\sigma = \sum_j \sigma_j \pmod{2^{HMAC_bitlength}}$.
   This is:

$$(SN_a; d_{1,a}, \ldots, d_{t,a}); \ldots; (SN_z; d_{1,z}, \ldots, d_{t,z}); \sigma$$

3. FN_i sends the aggregated tuple to its parent node.
   - If the parent node is U (thus, $FN_i = FN_b$), this is the final aggregated message.
   - Else, message aggregation is executed again at the parent node (another fog node).

4.3.4. User obtains the sensed data

U receives $(SN_a; d_{1,a}, \ldots, d_{t,a}); \ldots; (SN_z; d_{1,z}, \ldots, d_{t,z}); \sigma$ and obtains the sensed data by following these steps:

1. U computes the pseudo-random t-bit sequence $(c_{1,i}, \ldots, c_{t,i})$ that each SN_i built at Step 1 to encrypt their symbols. U uses the shared secret keys $K^v_{SN_i}$ and the random values v and p for this purpose.
2. U is then capable of retrieving the binary representation of the symbol sent by each SN_i as follows:

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

3. U computes the HMAC values that correspond to each symbol sent by each SN_i (Step 3), and aggregates all the computed HMACs as follows:

$$\omega = \sum_i (d_{1,i}, \ldots, d_{t,i} || p || v || K^v_{SN_i}) \pmod{2^{HMAC_bitlength}}$$

4. U checks whether $\omega = \sigma$.

- If the equality succeeds, U accepts the received symbols and sends an acknowledge to the orchestrated hierarchy by means of FN_b:

$$\{\text{“}Ack\text{”}||p||v, HMAC(\text{“}Ack\text{”}||p||v, K^v_{FN_b})\}$$

  FN_b verifies the HMAC of the received data, builds the message $\{\text{“}Ack\text{”}||p||v\}_{SK_{FN_b}}$ and multicasts it to the orchestrated hierarchy.

- Else, U does not accept the received symbols and she should run a protocol for suppressing compromised nodes from the orchestrated hierarchy.

  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 [27].

5. Security and privacy analysis

The security and privacy requirements of the proposed system are studied in this section. The discussion is organized in three propositions, and each proposition has several claims to support its fulfillment.

Our attacker model considers an adversary who can control fog nodes and sensor nodes (i.e, they become compromised nodes); and who can also access communication lines to capture, modify and retransmit messages. Moreover, 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 interests of the users if they have the chance.

Proposition 1. *The proposed system preserves the confidentiality, authenticity and integrity of the transmitted data in the presence of external and internal attackers that perform active and passive attacks.*

Claim 1. *An adversary cannot successfully eavesdrop transmissions during the orchestration process.*

During this protocol, the user sends to the bootstrap fog node a set of CP-ABE ciphertexts encrypted with the specific attributes of the requested service; these CP-ABE ciphertexts are next multicasted to all the hierarchy of nodes. These ciphertexts are generated by means of the CP-ABE public key PK^{cp-abe}, and only those entities with the right CP-ABE secret keys can

decrypt them. An external adversary does not have access to these secret keys and, hence, she cannot get any knowledge from capturing a CP-ABE protected transmission.

In the case of internal adversaries (i.e., a compromised fog or sensor node), they will only be able to decrypt the CP-ABE ciphertexts linked to the capabilities of the compromised node, hence, the information leaked will be very limited even in that challenging situation.

Claim 2. *An adversary cannot successfully eavesdrop transmissions during the service delivery process.*

During this protocol, the random challenge $p||v$ is multicasted through the network. This value provides no valid knowledge to external or internal attackers. As a response to the challenge, the sensed data is sent towards the user and aggregated at intermediate fog nodes. The sensed data are encrypted using temporal secret keys $K^v_{SN_i}$, which are only known by the user and each participant sensor SN_i. As a result, external or internal attackers will not be able to decrypt them and gather any information. In the case of a compromised sensor node, the adversary will learn only the data sensed by this device, but she will not be able to gather any additional information from the rest of the system.

Claim 3. *An adversary cannot successfully impersonate a certain fog node in front of other entities.*

Fog nodes use a temporal secret key $K^v_{FN_i}$ to build the HMACs of the exchanged messages with the users; this key is computed by means of the master key K_{FN_i}, which is only known by the fog node itself and the cloud. Regarding the messages multicasted by fog nodes to the sensor nodes, they are digitally signed using secret keys SK_{FN_b} that correspond to a public-key cryptosystem with signing capabilities. Only the fog node knows its signing secret key SK_{FN_b}. An adversary cannot get access to the required cryptographic material, and it is computationally unfeasible for her to tamper with digital signatures and HMACs.

Claim 4. *An adversary cannot successfully impersonate a certain sensor node in front of other entities.*

A sensor node uses a temporal secret key $K^v_{SN_i}$ to encrypt the sensed data and to build the HMAC that authenticate the transmission; temporal keys are computed by means of the master key K_{FN_i}, which is only known

by the sensor node itself and the cloud. An adversary cannot get access to the required cryptographic material, and it is computationally unfeasible for her to tamper with HMACs.

Claim 5. *An adversary cannot modify the messages sent by a user, a fog node or a sensor node without being detected.*

All the transmissions between entities are protected using HMACs or digital signatures. Secret keys for building HMACs are only known by the cloud and by each node individually; on the other hand, secret keys for building digital signatures are only known by each node individually. As a result, an external or internal adversary cannot modify the transmissions done by others without fully disrupting them; a situation that will be detected during the integrity verifications that the entities perform.

Claim 6. *An adversary cannot remove the contribution of a certain sensor node without being detected.*

The data sent by a certain sensor node SN_i can be erased at any level of the orchestrated hierarchy only if the HMAC σ_i linked to its contribution is known. If this value is not known, the adversary must guess the appropriate σ_i with a probability equivalent to the bit length of the HMAC algorithm. On the other hand, if this value is known and the contribution is erased, this fact will be considered as a non-transmittal by the user. Due to the *orchestration process* run by the user before starting the *service delivery process*, the user is aware of all the sensor nodes that should send their readings. As a result, a non-transmittal from a certain sensor node will allow the user to detect that an attacker has removed its contribution.

Claim 7. *An adversary controlling a compromised fog node cannot refuse to aggregate the data in transit without being detected.*

A compromised fog node may discard the messages received from some of its child nodes, refuse to aggregate them, and decide not to send the resulting data to its parent node. The user will consider this fact as a non-transmittal. In a similar way to Claim 6, due to the *orchestration process* run by the user before starting the *service delivery process*, the user is aware of all the sensor nodes that should send their readings and, hence, she will detect this attack.

Result 1. *As stated in the previous claims, the proposed system preserves the confidentiality, authenticity and integrity of the transmitted data in the presence of external and internal attackers. In the case of the internal attackers that have compromised fog or sensor nodes, i.e., the most challenging situation, their attacks will only leak the information known by the compromised nodes and the rest of the network will not be affected. Moreover, internal attackers trying to perform integrity attacks or refusing to aggregate the data in transit will be detected.*

Proposition 2. *The proposed system can only be used by authenticated and valid users, unless they compromise a fog node; in that case, the use will be limited.*

Claim 8. *The creation of a fake but valid authentication certificate is computationally unfeasible.*

A user willing to use the fog computing system is required to first perform the "User registration" step at the cloud. As a result of this process, the user and the cloud exchange digital certificates that will be used to authenticate themselves mutually in any further communication. An adversary trying to use the system must show her identity to the cloud and authenticate herself by means of a valid certificate. Generating a fake certificate that grants her access to the system is nowadays computationally unfeasible, without the knowledge of the secret key used by the certification authority.

Claim 9. *An adversary cannot exploit the different entry-points of the network to get the cryptographic material required to use the system without being authenticated.*

In order to use the system, an adversary must obtain the symmetric key $K^v_{FN_b} = H(v||K_{FN_b})$ to be shared by FN_b and the user of the service. Independently of whether an adversary tries to access the system by means of the cloud or by means of a nearby fog node, key $K^v_{FN_b}$ is only provided by the cloud that, before providing it, requires the requester to present her credentials and authenticate herself by means of an authentication certificate. According to Claim 8, the adversary is unable to generate this credential.

Claim 10. *An adversary cannot reuse past transmissions to repeat the execution of a past service.*

Valid users use the key $K^v_{FN_b}$ shared with the bootstrap fog node to build the HMACs of the exchanged messages. HMACs are built with a random and fresh value v that identifies the service. HMACs are verified by the fog nodes and, thus, reused messages containing past HMACs will be detected and discarded.

Claim 11. *An adversary who compromises a fog node can use the hierarchy or nodes rooted at the captured fog node.*

An adversary that captures a fog node FN_b and retrieves its secret key SK_{FN_b} can then multicast valid signed messages $\{p||v\}_{SK_{FN_b}}$ to the part of the network rooted at FN_b and then retrieve sensed data. Note that this does not affect the rest of the network.

Result 2. *As stated by the previous claims, a user must be properly authenticated to use the proposed system. Only in the case that the adversary compromises a fog node, she will be able to use a limited part of the network without the right credentials.*

Proposition 3. *The proposed system preserves the privacy of the users by ensuring that only the data that are strictly necessary to fulfill a service are revealed to the involved entities.*

Claim 12. *External or internal adversaries cannot obtain the detailed interests (e.g., type of sensed data or specific locations) of users during the orchestration process.*

During this process, users send a set of CP-ABE ciphertexts encrypted with the attributes linked to the type of sensed data or the specific locations from which the sensed data should be gathered. These ciphertexts-attributes follow a tree model (from more general to more specific) that is mapped to the hierarchy of devices that participate in the orchestrated hierarchy. As a result, fog nodes situated at upper layers of the orchestrated hierarchy can only know general attributes of the requested application if they are able to decrypt a ciphertext (i.e., "zone1" and "measurement"). While, on the other hand, sensor nodes situated at the bottom, are able to know specific attributes of the sensed data (i.e., "subzone1" and "temperature"), but not the whole request.

CP-ABE ciphertexts are computed with the public key of the CP-ABE cryptosystem and only the entities that know the right secret keys are able to decrypt them. External attackers do not have the required cryptographic

material and, hence, they cannot decrypt the CP-ABE ciphertexts and know the interests of the users. Regarding internal attackers who have compromised fog nodes or sensor nodes, their level of knowledge will depend on the number of nodes that they have compromised and on the hierarchy layer in which those devices are located.

Claim 13. *External or internal adversaries cannot obtain the detailed interests (e.g., type of sensed data or specific locations) of the users during the service delivery process.*

During this process, only the sensor nodes that can fulfill (part of) the required service according to their attributes will sense data and transmit them towards the user. If an adversary knows the specific attributes of the sensors that transmit data, she can then infer the interests of the user. Therefore, this situation would not leak any significant information unless the attacker is a very powerful adversary that has a complete knowledge of the whole fog computing system including all the sensor nodes involved (or a big part of them), their locations and sensing capabilities. We consider that an adversary with such powerful capabilities is rare and she is out of the adversary model considered in this paper.

Result 3. *As stated by the previous claims, the proposed system hides the specific attributes of the services requested by users which, in turn, hides their personal interests, thus, fulfilling the data minimization principle included in the EU GDPR. The proposed system is strong against external adversaries and regular internal adversaries capable of compromising only a limited number of nodes.*

6. Performance analysis

The proposed system is executed in an architecture made of fog nodes, sensor nodes and the cloud. Sensor nodes are lightweight devices and, for the system to be scalable, the part of the protocols to be executed by sensor nodes must work with a limited message length and an affordable computational cost.

There are four key aspects of the system that affect its scalability in terms of the effort required at the sensor nodes to process the message length and perform the required computations. These elements are defined next:

- NU: Number of users who use the system at the same time.

- $NF + NS$: Number of fog and sensor nodes present in the system.
- NA: Number of attributes that define the capabilities of fog and sensor nodes.
- HL_i: Number of generalizations (i.e., hierarchical depth) of each attribute i.

In the next two subsections we analyze how these aspects affect the scalability of the proposed system when running the two proposed protocols: *secure orchestration* and *secure service delivery*. In the last subsection, we discuss the results of the analysis and provide some empirical figures.

6.1. Scalability of "Protocol-1: Secure orchestration"

In this protocol, a user U generates the list of policies that apply to her requested service, where there are as many policies as combinations of attribute values and generalizations that apply to the aforementioned service. Next, U computes one CP-ABE ciphertext by each policy from the list. The number of CP-ABE ciphertexts depends on the number of attributes involved in the request (NA, in the worst case) and the number of generalizations of each one (HL_i). As a result, the total number of CP-ABE ciphertexts that U generates can be computed as: $\prod_{i=1}^{NA} HL_i$. In the worst case (i.e., there is only one fog node between the user and the sensor nodes), the whole list of ciphertexts minus one will reach the sensor nodes.

According to that, sensor nodes, for each user who performs an orchestration, are required to verify a digital signature (computed by the boostrap fog node), decrypt $(\prod_{i=1}^{NA} HL_i) - 1$ CP-ABE ciphertexts, and compute an HMAC. This implies a total *computational cost* of $O(NU) * (O(1) + O(HL^{NA}) + O(1))$ and a message length that mainly grows according to $O(NU) * O(HL^{NA})$. It can be seen that the main aspects that affect the scalability of the system in this protocol are NU, HL and NA.

In theory, the number of possible different attributes NA is the most relevant one due to the fact that it would produce an exponential cost if this element grows. Fortunately, NA directly depends on the different types of sensing capabilities of the sensors deployed in the system, which is assumed to be a limited and fixed number. Moreover, the actual service requests would usually involve a small subset of the NA attributes, so that the number of policies and ciphertexts required will be exponentially lower than $(\prod_{i=1}^{NA} HL_i) - 1$. So we argue that in a real deployment of the proposed system, value NA will not significantly affect the scalability of the proposal.

Element HL_x depends on the depth of the hierarchy of fog and sensor nodes that, in turn, depends on the size of the network and the specific deployment of the different nodes. In any case, HL_x is a fixed value and, therefore, its practical effect in the scalability of the system is limited.

Finally, the number NU has a significant impact on the scalability of this protocol due to the fact that the number of computations and the size of the messages to be processed by sensor nodes grow linearly with the number of users using the system simultaneously. Nevertheless, the aforementioned cost is linear, and the orchestration process focuses only on a specific part of the network. This means that if the network is large, it can accommodate a large number of users simultaneously using different sets of nodes. If the network is small, there should be the option of limiting the number of simultaneous users in the system, an option that may be enforced by the cloud during its access control step.

6.2. Scalability of "Protocol-2: Secure service delivery"

In this protocol, a user U sends a challenge $\{p||v\}$ which is digitally signed and multicasted to the sensor nodes by a bootstrap fog node. Therefore, the nodes receive a message with fixed length and they have to verify one digital signature. Next, sensor nodes build a message that contains their sensed value and that is sent to their parents in the fog computing hierarchy. This message is built as the tuple: $(SN_i; d_1, \ldots, d_t; \sigma_i)$, in which: i) SN_i is a fixed integer that identifies the sensor node; ii) $d_1, \ldots, d_t$ is a t-bit vector that encodes the sensed value and which has been computed by means of low-cost bitwise operations; and iii) σ_i is the output of an HMAC operation, a low-cost computation that generates a fixed length result.

Fixing value t to accommodate the bit-length of the sensor readings to be sent, the total length of the messages in this protocol are constant $O(1)$. Regarding the computation cost, all the operations involved in this protocol have a cost of $O(1)$ and they take negligible time. As a result, none of the aspects considered below have an effect on this protocol, and we consider it to be scalable.

6.3. Discussion and empirical results

The individual analysis of both protocols from the scalability point of view reflects that, as expected, the *secure orchestration* protocol is significantly more costly than the *secure service delivery* protocol. In particular, the *secure orchestration* protocol requires the use of expensive ABE and its scalability may be affected by the number of attributes and by the number of simultaneous users in the system. On the other hand, the *service delivery*

protocol is scalable and all the cryptographic operations involved (i.e., bitwise, HMAC and digital signature verification) are lightweight; in this sense, note that the verification of digital signatures built with the Elliptic Curve Digital Signature Algorithm[7]) is reported in the literature as affordable for IoT devices [28].

The issues detected in the *secure orchestration* protocol were expected and, in fact, they are shared with other works in the literature, such as [29, 19, 30], which use ABE in lightweight devices. In contrast with them, in our proposal, we concentrate the use of this expensive cryptography only on the orchestration of the service, which is executed just once for a whole service request; while each service delivery is run by means of the efficient and scalable *secure service delivery* protocol. Due to the fact that the number of runs of the *service delivery* protocol is expected to be much larger than the number of runs of the *secure orchestration* protocol, we argue that the proposed system represents an improvement in terms of scalability and use of resources with respect to the related works that tackle the same scenario.

To complement the theoretical analysis, we next provide empirical values for the three main cryptographic operations that sensor nodes must perform: i) decryption of CP-ABE ciphertexts; ii) verification of digital signatures; and iii) computation of HMACs.

We have tested those operations in a *Raspberry Pi 1 Model B* device. The basic 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*[8] as operating system, and the operations have been implemented using *Python*[9] as programming language and the cryptographic libraries: *charm*[10] and *pycrypto*[11].

The charm library provides an implementation of the CP-ABE approach presented in [25] and an implementation of the Elliptic Curve DSA (ECDSA). The pycrypto library provides implementations for computing HMACs and verifying DSA signatures. In any case, our system is agnostic of the specific implementation of the cryptographic primitives and can accommodate (and benefit from) more efficient implementations available now or in the future.

Table 1 summarizes the runtimes (in seconds) obtained for each opera-

[7] http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
[8] https://www.raspberrypi.org/downloads/raspbian/
[9] https://www.python.org/
[10] https://github.com/JHUISI/charm
[11] https://github.com/dlitz/pycrypto

Table 1: Computational cost for different cryptographic operations tested in a *Raspberry Pi 1 Model B*

Attribute based encryption operations (orchestration protocol)	Cost
CP-ABE decryption (2 attributes policy)	0.3202s
CP-ABE decryption (4 attributes policy)	0.5511s
CP-ABE decryption (8 attributes policy)	1.0100s
CP-ABE decryption (10 attributes policy)	1.2518s
CP-ABE decryption (15 attributes policy)	1.8265s
CP-ABE decryption (20 attributes policy)	2.4042s
Common cryptographic operations (service delivery protocol)	**Cost**
DSA signature verification	0.0720s
ECDSA signature verification	0.0171s
HMAC generation	0.0004s

tion. The provided numbers are the average of 100 runs. We classify those operations in *attribute based encryption operations*, run only by the secure orchestration protocol; and *standard cryptographic operations*, run by the secure service delivery protocol. As expected, the standard cryptographic operations take negligible time to be completed and are fully suitable for lightweight devices. On the other hand, CP-ABE decryption is a clearly expensive operation; however, the actual run time figures are reasonably low even when considering a large number of attributes. This point, together with the fact that the secure orchestration protocol is expected to be run a small number of times, allow us to conclude that our proposal is expected to be feasible for lightweight devices and, in comparison with related works, will result in significantly reduced latencies and energy consumption during the service delivery.

7. Concluding remarks and future work

In this paper, we have proposed a secure and privacy-by-design system for orchestrating and delivering fog-enabled IoT services. As far as we know, this constitutes the first practical solution to fog orchestration that carefully considers both privacy and security issues. Moreover, this differentiates our approach from other systems that enforce security by employing expensive cryptography during whole life cycle of the service management. In contrast, our approach provides a more efficient solution that, thanks to orchestration,

suffices to use standard –efficient– cryptography during the service delivery and makes a more rational use of the fog resources.

In addition to that, we assumed a challenging scenario in which the entities in the fog computing system exchange data in open and unsecured networks and can be subjected to a wide range of passive and active attacks, which include compromising and controlling some devices. The security and scalability of our protocols have been carefully analyzed; the conclusions and results of such analysis reflected that our proposal is secure and feasible.

Regarding future work, the presented proposal is designed to work under a pure request-response paradigm. A first line of further research would be to adapt the proposed system to deal with continuous streams of data; in this way, we would cover a wider range of fog computing-based IoT applications. Finally, we also plan to implement a full prototype of the proposed system and evaluate its capabilities in a relevant environment such as a high-fidelity laboratory environment or a simulated operational environment (i.e., level 6 in the *Technology Readiness Levels*[12] scale).

Acknowledgments and disclaimer

This work was partly supported by the European Commission (project H2020-700540 CANVAS), by the Spanish Government (TIN2014-57364-C2-R SmartGlacis, TIN2016-80250-R Sec-MCloud and TIN2015-70054-REDC), and by the Spanish DGT (SPIP2017-02250 ISSUM). 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] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, 2012, pp. 13–16.

[2] 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.

[12] https://www.acq.osd.mil/chieftechnologist/publications/docs/TRA2011.pdf

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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) E1695.

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

[26] D. Chakraborty, A. Joshi, Y. Yesha, T. Finin, Toward distributed service discovery in pervasive computing environments, IEEE Transactions on Mobile Computing 5 (2) (2006) 97–112.

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

[28] E.-O. Blaß, M. Zitterbart, Towards acceptable public-key encryption in sensor networks, in: Proceedings of the ACM 2nd International Workshop on Ubiquitous Computing, 2005, pp. 88–93.

[29] 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.

[30] 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.