International Journal of Information Security (2021) 20:695–713
https://doi.org/10.1007/s10207-020-00526-3

REGULAR CONTRIBUTION

Attribute-based encryption and sticky policies for data access control in a smart home scenario: a comparison on networked smart object middleware

Sabrina Sicari[1] · Alessandra Rizzardi[1] · Gianluca Dini[2] · Pericle Perazzo[2] · Michele La Manna[3] · Alberto Coen-Porisini[1]

Published online: 23 November 2020

Abstract
Regulating the access to the Internet of Things (IoT) network's resources is a complex-prone task, which requires to pay a great attention on how policies are defined, shared, and enforced. The present paper considers the specific context of a smart home, which represents one of the main IoT application domains, and it focuses on two solutions proposed in the literature to cope with the aforementioned issues. On the one side, approaches based on attribute-based encryption (ABE) allow one to encrypt data for multiple recipients, in such a way that only those recipients whose attributes satisfy a given access policy can decrypt afterward. ABE guarantees a high level of customization due to the variety of attributes which can be defined, and it is also flexible enough to be adapted to different kinds of scenarios. On the other side, approaches based on sticky policies allow to attach an access policy directly to the data itself, and to employ a trusted authority to evaluate and enforce the policy itself. Sticky policies also guarantee a highly distributed and customizable enforcement of access control rules. In this paper, we compare the advantages and the drawbacks in terms of performance and robustness of such two techniques by means of their integration within the prototype of an IoT middleware, named networked smart object. Hence, the effectiveness of the presented solutions is validated by means of a real test-bed in the smart home scenario, in terms of storage occupancy, CPU load, and data retrieval delay. The final goal is to reveal the best approach to be used depending on the application's requirements.

Keywords Internet of Things · Security · Attribute-based encryption · Sticky policy · Access control · Middleware

1 Introduction

The spreading and continuous development of Internet of Things (IoT) technologies and services introduces a new way of conceiving and managing the information transmitted over the network [1]. The huge amount of data generated and shared every second is in constant increment, thus raising significant scalability issues. One reason for the success of the IoT paradigm is certainly the introduction of miniaturized devices, which are able to interact and acquire information from the environment where they are placed in. Besides such a perk, those devices are often memory- and energy-

✉ Sabrina Sicari
sabrina.sicari@uninsubria.it

Alessandra Rizzardi
alessandra.rizzardi@uninsubria.it

Gianluca Dini
gianluca.dini@ing.unipi.it

Pericle Perazzo
pericle.perazzo@ing.unipi.it

Michele La Manna
michele.lamanna@unifi.it

Alberto Coen-Porisini
alberto.coenporisini@uninsubria.it

1 Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, Via O. Rossi 9, 21100 Varese, Italy

2 Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy

3 Università di Firenze, Via Santa Marta 3, 50121 Florence, Italy

constrained and, as such, they have a low capability to handle complex data processing and heavy security tasks by themselves.

An important issue to be addressed is how the information acquired by such devices, which act as producers, could be shared with the interested consumers. In fact, in the IoT context, multiple parties may be involved, thus requiring the definition of strict rules for regulating the access to the IoT resources. In particular, sensitive data must be disclosed only to authorized parties. Infrastructures, both public and private, that make use of IoT technologies could grow faster by ensuring their customers the reliability and the trustworthiness of their data management practices.

In such a direction, two different approaches seem to be promising in providing an effective solution to the aforementioned issues. The first approach involves a cryptographic technique called attribute-based encryption (ABE) [2]. ABE is an encryption technique that allows only those who comply with a given access policy to decrypt the desired information. Such an access policy is defined in terms of attributes of the decryptor or of the data itself. Though more energy consumptive than traditional symmetric or asymmetric cryptography, ABE is very powerful as it allows to make data safely rest or travel over untrusted channels and platforms, and, at the same time, enforce a fine-grained access control.

The second approach involves the use of sticky policies [3], which can be defined by the producer, and can travel along with the associated information through the whole data life cycle. Recipients are allowed to retrieve the desired information only according to the associated sticky policy, which is evaluated by a trusted authority. Though sticky policies require the trusted authority to be always online in order to provide the required decryption keys, it can be very lightweight as it can exclusively leverage symmetric cryptography.

In this paper, ABE and sticky policies techniques are compared, in order to reveal their advantages or drawbacks in a smart home scenario, with respect to robustness in terms of reliability and performance (e.g., storage occupancy, CPU load, data retrieval delay). The main goal behind this work is to establish the differences in choosing one of the two approaches with respect to certain application domain's requirements. To this end, both the approaches have been integrated within the same existing flexible and cross-domain middleware, named *networked smart object (NOS)*. NOS is an IoT platform, originally conceived to manage data generated by heterogeneous sources, and share them with interested parties, adopting specific algorithms and protocols [4]. Note that an enforcement framework based on sticky policies is already available for the NOS architecture, as presented in [5]. Instead, in this paper, a specific type of ABE, named cipher-text-policy attribute-based encryption (CP-ABE) [6], has been integrated within the NOS system for comparison purpose. CP-ABE has been also considered due to its similarities with the approach based on sticky policies, as clarified later in the paper. To give some preliminary details, note that, in the CP-ABE paradigm, as for sticky policies, the access rule resides within the encrypted data itself, while the attributes, used for evaluating the policy, are directly associated with decryptors.

Summarizing, the main contributions proposed in this work are the following ones:

– CP-ABE scheme has been integrated within NOS architecture. Note that NOS platform has been chosen due to its modular architecture, which enables to dynamically adapt its behavior; thus, it is particularly suitable to be extended with new functionalities. Moreover, an implementation of sticky policies in NOS platform already exists [5].
– the behavior of CP-ABE and sticky policies approaches has been compared, in order to reveal their potentialities and weaknesses from a functional point of view, following the data flow management in a not-fully trusted environment, as it happens in the typical IoT contexts. The comparison is made in a smart home scenario, where different IoT devices produce heterogeneous data, from video streams to electrical data sets. Such a kind of scenario also enables the presence of different kinds of users. In this way, a simple yet accurate case study is provided, which could be further expanded for future analysis.
– the performance of the employed CP-ABE scheme with respect to the sticky policy method has been evaluated, by means of a real test-bed. The aforementioned smart home data set is considered, and the following metrics are analyzed and measured: storage occupancy, CPU load, and data retrieval delay. The main outcomes reveal that the sticky policy approach is more efficient in terms of CPU load, but its storage occupancy and data retrieval delay are higher than those of the CP-ABE approach.

The remainder of the paper is structured as follows. Section 2 presents the preliminaries of the proposed work, which include the sticky policy and CP-ABE paradigms, and the basic NOS architecture. Then, Sect. 3 presents the integration of CP-ABE functionalities into the NOS middleware, along with the comparison between the CP-ABE approach and the one based on sticky policies. Section 4 presents the threat model, the smart home application scenario, and the performed experiments. In Sect. 5, the related work is presented, while Sect. 6 ends the paper, also drawing some hints for future works.

2 Preliminaries

In this section, the necessary preliminaries for clearly understanding the mechanisms related to the adoption of sticky policies and CP-ABE for securing the access to the information transmitted within an IoT system are detailed. Moreover, a sketch of NOS architecture is presented.

2.1 Networked smart objects architecture

Two main entities compose a typical IoT system: (i) the data producers, conceived as heterogeneous data sources (e.g., WSN, RFID, NFC, actuators, etc.) which generate data to be sent to the IoT platform; (ii) the data consumers, who interact with the IoT platform through services making use of such IoT-generated data, typically accessing them by means of a mobile device (e.g., smart phone, tablet) connected to the Internet, through WiFi, 3G, or Bluetooth technologies.

In such a scenario, networked smart objects' (NOS) middleware [4] has been conceived as a layered architecture, providing lightweight and flexible functionalities; it really represents a comprehensive approach for managing data gathered from heterogeneous sources in a distributed way, and for providing customized services to users, assessing security as well as data quality requirements. It is worth to remark that NOS, according to authors' knowledge, still represents the unique architecture, available in the literature, able to address both security and data quality issues.

Proper interfaces for the communications of NOSs with the data producers and consumers have been defined. HTTP protocol is usually adopted for collecting data from the IoT devices. For each incoming data, the following pieces of information are gathered:

- the kind of data producer, which describes the type of IoT source;
- the communication mode, that is, the way in which the data is collected (e.g., discrete or streaming communication);
- the data schema, which represents the type (e.g., number, text) and the format of the received data;
- the data content;
- the reception timestamp.

Since the received data are of different types and formats, NOSs initially put it in the *Raw Data* storage unit. Data in such a collection are periodically processed, in a batch way, in the *Data Normalization* and *Security and Data Quality Analysis* phases, in order to obtain a uniform representation and add useful metadata regarding security (i.e., level of confidentiality, integrity, privacy and robustness of the authentication mechanism) and data quality (i.e., level of accuracy, precision, timeliness and completeness) assessment. Such an assessment is performed following well-defined algorithms, which are detailed in [4]. It allows the consumers, who access the IoT data, to be aware of the levels of reliability and trustworthiness of the services gathered by NOSs themselves. Hence, consumers can directly filter by themselves the data processed by NOSs, according to their personal preferences, in terms of security and quality.

Instead, message queue telemetry transport (MQTT) protocol [7] is used for disseminating the information to the interested data consumers. To this end, a topic is assigned by NOSs to each processed data. NOSs also provide a lightweight and secure information exchange process, based on an authenticated publish and subscribe mechanism [8], integrated with the aforementioned MQTT protocol.

Finally, it is worth to remark that NOSs modules interact among themselves through *RESTful* interfaces; they have been implemented in a real prototype [4]. *Node.JS* platform [9] has been used for developing NOSs' core operations, *MongoDB* [10] has been adopted for the data management, and Mosquitto [11] has been chosen for realizing the open-source MQTT broker. For more details about the implementation, please refer to [4].

A scheme of NOS architecture is sketched in Fig. 1, along with its current integration with sticky policy enforcement framework [5], which is described in the following section.

2.2 Sticky policies

The sticky policy paradigm was first proposed by Karjoth, Schunter, and Waidner [12]. Sticky policies are transmitted along the data they refer to throughout the entire data life cycle. Specifically, sticky policies allow us to define the following aspects:

- the owner of the data;
- the data content, possibly encrypted;
- the scope of the data;
- where and when data will be available;
- specific obligations and restrictions.

In detail, the concept of sticky policy is to attach security and privacy policies to owners' data and drive access control decisions and policy enforcement. Sticky policies allow specifying access rules in an extremely fine-grained manner: in principle, every data unit could have its own, unique, policy. Furthermore, as policies 'travel' with the data across the entire system, they could provide protection over the entire data life cycle. Such an approach has been mainly introduced for security and privacy enforcement: when submitting data to a consumer, a user consents to the applicable policies selecting the proper preferences.

Such features are particularly interesting in some scenarios, as that of IoT, where users' or business' confidential

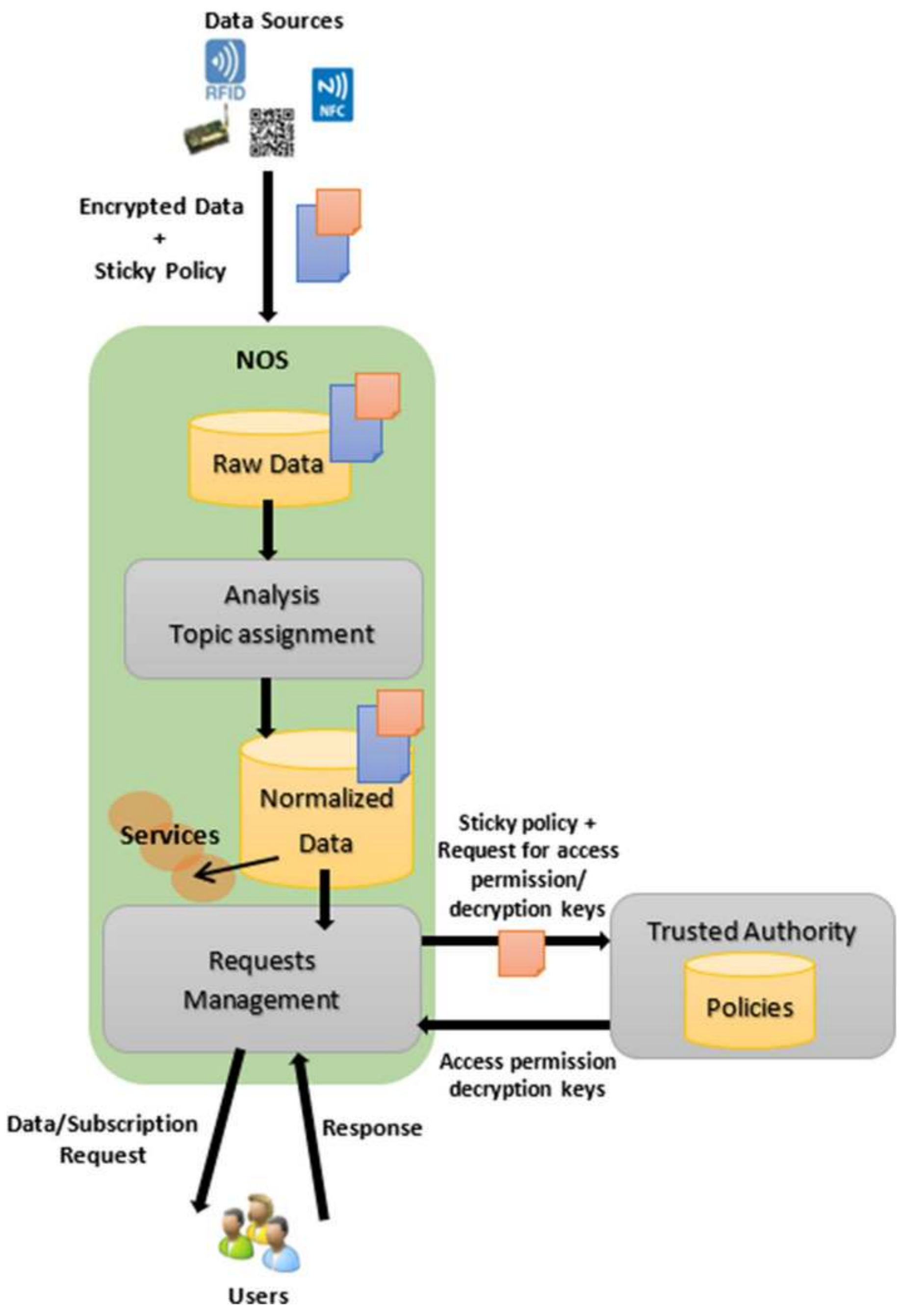
Data Sources
RFID
NFC
Encrypted Data + Sticky Policy
NOS
Raw Data
Analysis Topic assignment
Normalized Data
Services
Requests Management
Sticky policy + Request for access permission/ decryption keys
Trusted Authority
Policies
Access permission decryption keys
Data/Subscription Request
Response
Users

Fig. 1 NOS data flow with sticky policies

information may flow across organizational boundaries [3]. For example, social networks may share some information with marketing companies; similarly, cloud applications may transfer data, depending on a need, among different realms. Such situations represent well-known open issues in the field of security and privacy enforcement.

The sticky policy concept has already been integrated in the NOS platform, as presented in [5]. Note that NOSs own no policies/credentials, because an external *Trusted Authority* (TA) is responsible for their management. The owner of the data sends them in an encrypted way along with the associated sticky policy to NOS; clearly, data producers have an in-depth control over the flow of their own information, since they are responsible for the access rules on their own data. Then, each NOS can contact the TA in order to obtain the access permissions on the received data, when there is the need to disclose them to interested consumers (i.e., the users who interact with the IoT platform). In this way, no synchronization or policy sharing is required among multiple NOSs, since the access permissions are managed by the TA. Figure 1 summarizes the just described behavior, as anticipated in Sect. 2.1. Instead, the CP-ABE approach, that up until now has not been integrated within NOS platform, will be introduced in the following section.

2.3 Cipher-text-policy attribute-based encryption

Attribute-based encryption (ABE) [2] is a cryptographic technique which allows one to encrypt data in such a way that only the parties compliant with a given access policy can decrypt it afterward. Access policies are Boolean formulas, defined on some attributes, which describe the interested consumer or the encrypted data itself.

Two ABE paradigms are available in the literature: cipher-text-policy attribute-based encryption (CP-ABE) [6] and key-policy attribute-based encryption (KP-ABE) [13]. With CP-ABE, each party owns a decryption key generated with a set of attributes which describes him/her. Who encrypts the data determines the access policy to be used to decrypt it. Decryption is possible if and only if the attribute set of a decryption key satisfies the policy embedded in the encrypted data, whereas with KP-ABE the mechanism is the opposite: each party owns a decryption key generated from a policy that determines which kind of data he/she can access. Data, instead, are encrypted under a list of attributes which describe the underlying information. As emerged, CP-ABE more resembles the sticky policy principles because the access policy travels with the data. Hence, this work is focused on the CP-ABE paradigm, instead of the KP-ABE one.

The first CP-ABE scheme was proposed by Bethencourt, Sahai and Waters in [6]. It represents a public-key encryption scheme based on bilinear pairing. Basically, all CP-ABE attributes can be intended as Boolean ones, in the sense that the presence of a given attribute inside a decryption key counts as a "true" in the policies that contain such an attribute, and its absence counts as a "false." In [6], a method to implement numerical attributes by means of multiple Boolean attributes is introduced. A numerical attribute is realized by means of its binary representation on a fixed number of bits. Two Boolean attributes are used for each bit: one of them mapping the condition that the relative bit is zero, and the other that the relative bit is one. By doing so, it is possible to efficiently realize access policies including comparison operators (e.g., $=, \neq, <, \leq, >, \geq$) between a numerical attribute and a constant. In order to ease the reading, in this paper the choice is to abstract away from the mathematical details and focus on the application programming interface. The interested reader can refer to [6] for more details. Hence, the discussion reported hereby concerns the main functions of the CP-ABE mechanism.

Note that, in general, the mechanisms based on ABE needs a *Trusted Authority* (TA) to setup and managing some functions of the system, as happened for sticky policies in Sect. 2.2; in particular, the TA is in charge of:

- generating and distributing the key used for encryption, called *encryption key* EK, which is unique for the whole system;
- generating and assigning each data consumer a *decryption key* DK.

In the following, an attribute set is denoted by the symbol γ, while a policy is denoted by the symbol $\mathcal{T}$. The CP-ABE scheme is modeled by the following black-box primitives:

$$(\mathrm{MK}, \mathrm{EK}) = \mathrm{Setup}(\kappa) \tag{1}$$

This primitive initializes the CP-ABE scheme. It takes as input the security parameter κ, and it outputs a master key MK, which is kept secret by the TA, and an associated encryption key EK, which is publicly divulged. The Setup primitive is executed by the TA.

$$C = \mathrm{Encrypt}(M, \mathcal{T}, \mathrm{EK}) \tag{2}$$

This primitive encrypts a plain-text M with the policy $\mathcal{T}$. It takes as input the encryption key, EK, and it outputs the encrypted data C, which embeds the policy $\mathcal{T}$. The Encrypt primitive can be executed by any component of the IoT network, because it does not require the knowledge of any secret. It is worth to note that this primitive is computationally demanding, so making its execution challenging for a resource-constrained IoT device [14,15]. This is a reason for adopting an architecture like the NOS one, as detailed in Sect. 3.

$$\mathrm{DK} = \mathrm{KeyGen}(\mathrm{MK}, \gamma) \tag{3}$$

This primitive generates a decryption key DK for a data consumer, described by the attribute set γ. It takes as input the master key MK and a set of attributes γ, and it outputs a decryption key DK, which embeds γ. The KeyGen primitive is executed by the TA.

$$M = \mathrm{Decrypt}(C, \mathrm{DK}) \tag{4}$$

This primitive takes an encrypted data C and a decryption key DK as input, and it outputs the plain-text message content if the interested consumer can access the data; otherwise, the decryption is unsuccessful and the primitive outputs nothing. Inside C, there is the policy $\mathcal{T}$. By the mathematical properties of CP-ABE scheme, decryption is successful only if the attribute set γ embedded in DK satisfies the policy $\mathcal{T}$ embedded in C. It is worth to remark that a policy is a Boolean formula, composed by certain attributes. If an attribute mentioned inside a policy belongs to γ, it is considered as "true" for the policy evaluation. So, a policy $\mathcal{T}$ is satisfied by an attribute set γ if the Boolean formula represented by $\mathcal{T}$,

Table 1 Acronyms

Acronym	Meaning
EK	The encryption key
DK	A decryption key
γ	An attribute set
$\mathcal{T}$	A policy
MK	The master key
M	A plain-text
C	A cipher-text
AVL	Attribute version list
CAL	Consumers attribute list
TA	Trusted authority

evaluated with the attributes γ, returns "true." The Decrypt primitive is executed by a data consumer holding the proper decryption key. It is worth to note that this primitive is, in general, computationally demanding for resource-constrained devices, but it can be easily executed by modern mobile devices such as smart phones and tablets, as proven in [16].

Table 1 summarizes the acronyms just presented and others, which will be used later.

3 Integration of CP-ABE into NOS architecture and comparison with sticky policies

Generally, an enforcement framework is composed by the following main standard elements: (i) a *Policy Enforcement Point (PEP)*, which intercepts the access requests and queries the PDP about its acceptance; (ii) a *Policy Decision Point (PDP)*, which evaluates the access requests against the authorization policies and takes the authorization decisions; (iii) a *Policy Administration Point (PAP)*, which contains the full set of authorization policies established by the system's administrators. In a previous NOS's version, such components are all located into NOS [17].

By introducing the sticky policies (Sect. 2.2), only the PEP is located into NOSs, while the PDP is located within the TA, as the PAP. As a consequence, the role of NOSs in the enforcement process is softened, and NOSs can be no longer considered as a single point of failure in the security of the information transmitted within the whole IoT system. By delegating some operations and controls to the TA, the overall efficiency of the NOSs middleware has been improved, as demonstrated in [5], with respect to the more traditional solution, presented in [17].

However, the main drawback emerged from the approach based on sticky policies is the need of an always-on-line TA, responsible of trustworthy evaluating the policies in relation to the subscribing data consumer. In order to overcome such an issue and, at the same time, to provide a flexible and efficient data access control framework, CP-ABE scheme is introduced. It includes a mechanism for access control able to embed PEP and PDP just inside the cipher-text (hence, the policy is not separated from the encrypted data itself, as it is for sticky policies). In order to include the new components/functionalities required by the CP-ABE scheme, presented in Sect. 2.3, the actual NOS architecture, described in Sect. 2.1, must be revised.

In Fig. 2, the modified NOS architecture is shown, including the CP-ABE primitives, which are integrated into the data flow. More in detail, with respect to Fig. 1, it worth to note that the policy associated with the data now depends on the CP-ABE encryption (the Encrypt primitive is executed by the new introduced *CP-ABE Data Encryption* module). The *CP-ABE Data Encryption* module has three main goals: (i) it performs the encryption in place of data producers, thus lightening the memory- and energy-constrained IoT devices from such a complex and expensive task; (ii) it defines and properly associates the access policies with the processed data; (iii) it stores normalized data in the *Normalized Data* storage unit in an encrypted form, so such data are protected also in the case of NOS compromised by unauthorized parties. The *CP-ABE Data Encryption* module combines the policies defined by NOS and the encryption keys defined by the TA, which executes the Setup and KeyGen primitives. In this sense, data owners have less control on the disclosure of their own information, but the IoT platform has more control on them. Another fundamental remark is that the TA now needs to communicate with both NOS, in order to provide the required encryption key for performing the encryption task, and users, to disclose the decryption key to authorize them to access the NOS's resources; hence, the Decrypt primitive is executed by the data consumer. After that, the TA can go offline.

Going more deeply into the differences between the NOS' data flow with the sticky policies approach and with the CP-ABE scheme, a further overview of the two systems is provided in Figs. 3 and 4, respectively.

Figure 3 highlights that, in the sticky policies approach, for each subscription to a certain topic by an interested consumer (step 10), the TA must be contacted by NOS in order to achieve the access decision (steps 11-13); then, if the TA agrees to the subscription, the data consumer will be notified of the data belonging to the requested topic (step 14). The credentials used for exchanging such data between NOS and the consumer are established *a priori* by an agreement between them (steps 1–2). Hence, a sort of double agreement should be done: one for the encryption key and another one for the topic's subscription. The normal NOS's processing activity is independent from such tasks (steps 3-9).

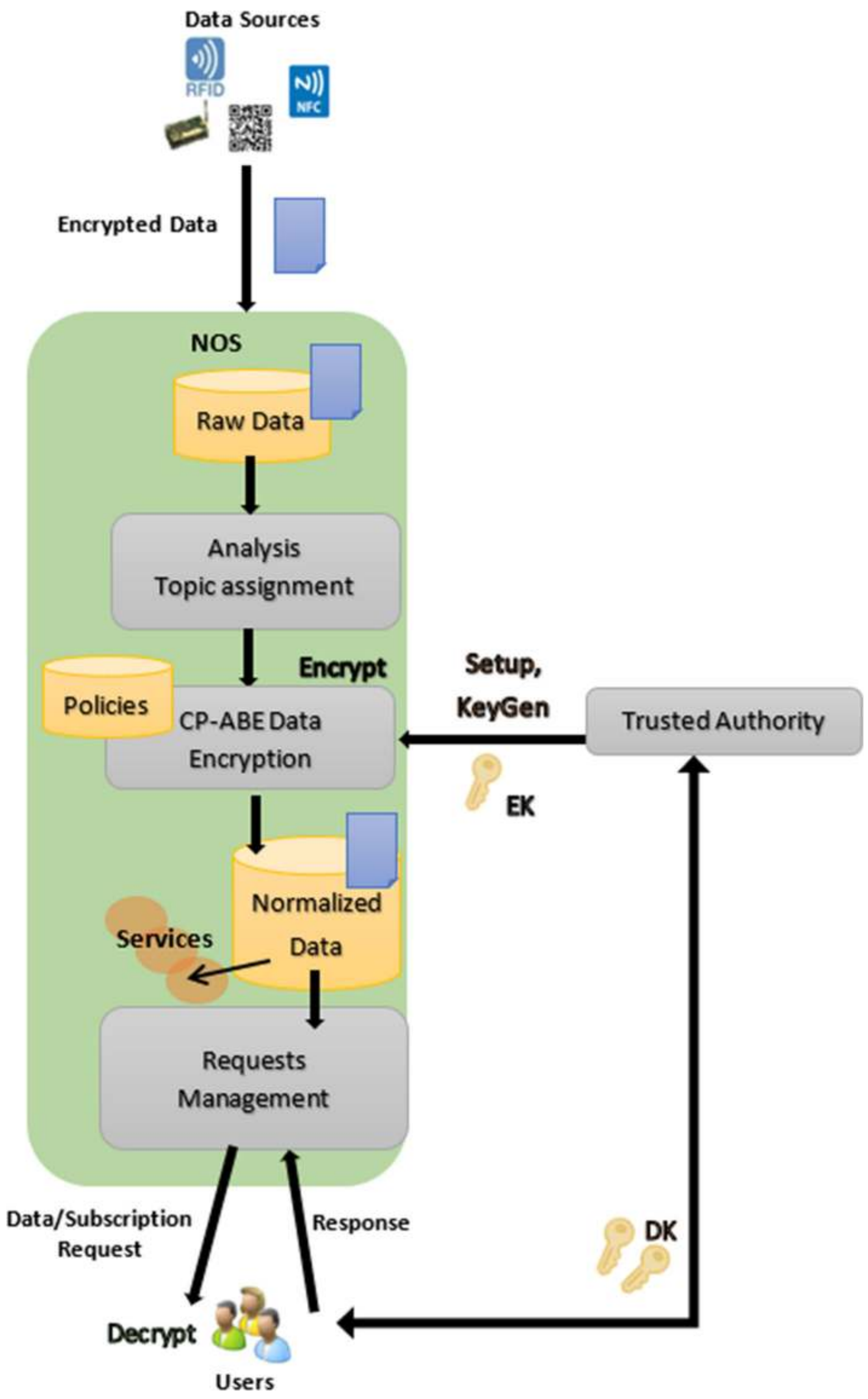

Fig. 2 New proposed NOS architecture

Instead, in a scenario which adopts CP-ABE (Fig. 4), the TA is no longer required to be online during subscription and data transmission (steps 8–9). NOS is not required to know the decryption keys needed by the consumers to decrypt the information (step 10). This lightens the keys' management from the NOS's viewpoint. In fact, with CP-ABE, the access policy is just embedded in the encrypted data on the basis of the assigned attributes (steps 1–7). Such an aspect represents the crucial difference between the two approaches, which clearly reveals the effectiveness of the CP-ABE approach in facilitating, from a performance perspective, the whole management of data encryption/decryption in relation to the established policies. In that sense, sticky policies seem more

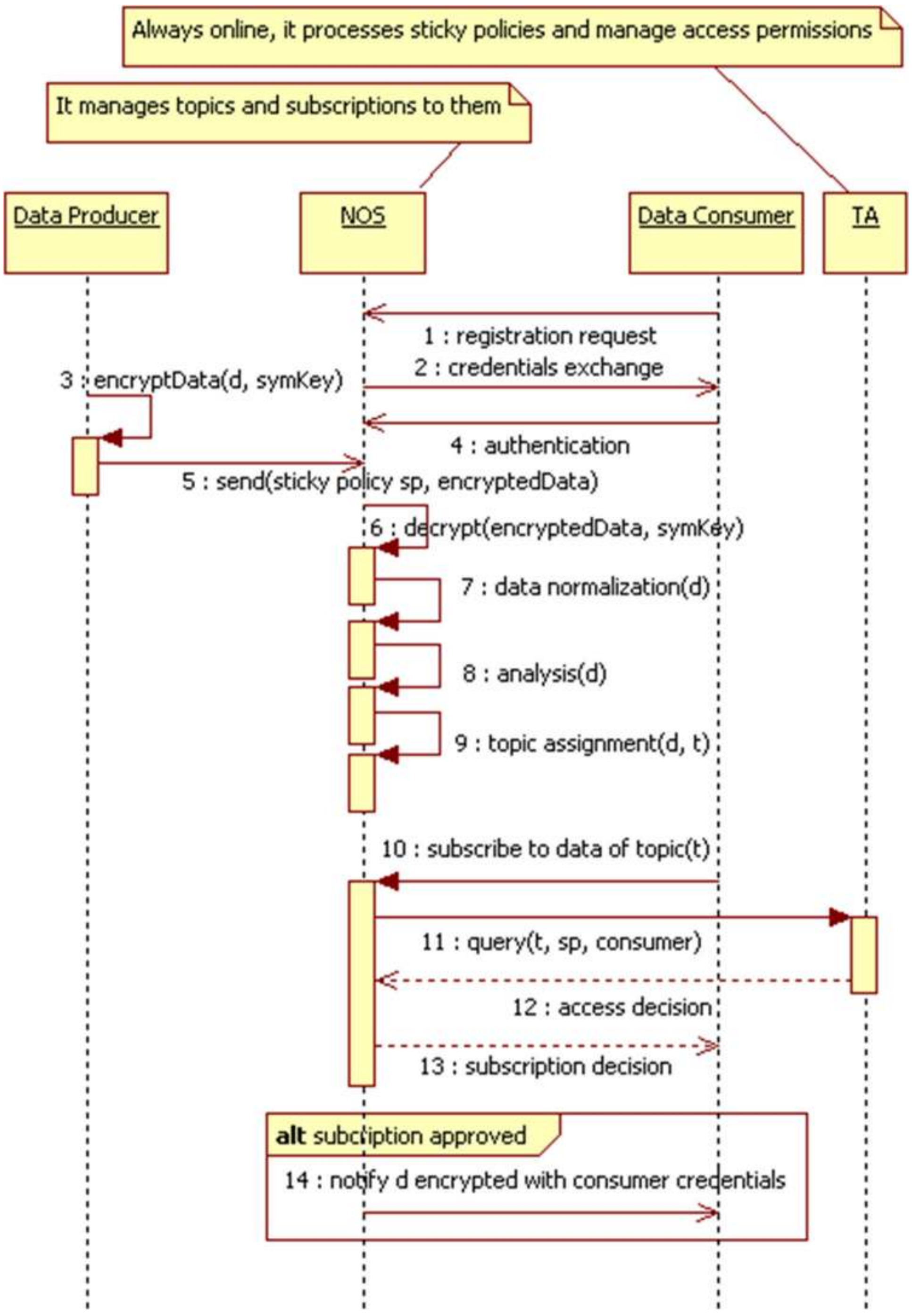
Always online, it processes sticky policies and manage access permissions
It manages topics and subscriptions to them
Data Producer
NOS
Data Consumer
TA
1 : registration request
2 : credentials exchange
3 : encryptData(d, symKey)
4 : authentication
5 : send(sticky policy sp, encryptedData)
6 : decrypt(encryptedData, symKey)
7 : data normalization(d)
8 : analysis(d)
9 : topic assignment(d, t)
10 : subscribe to data of topic(t)
11 : query(t, sp, consumer)
12 : access decision
13 : subscription decision
alt subcription approved
14 : notify d encrypted with consumer credentials

Fig. 3 Scheme of sticky policies-based data flow within the NOS system

difficult to manage because encryption/decryption are not so related to access policies with respect to CP-ABE paradigm.

In case of policies' update, addition, or revocation, the sticky policy approach requires an update of the scopes and constraints of the TA; hence, the related decryption keys must be revoked and re-assigned to the consumers involved in that policies. On the other hand, CP-ABE mechanism simply requires that NOSs change the policy used for encrypting data, which can be done without involving the TA.

Summarizing, the CP-ABE Encrypt, Setup and KeyGen primitives are not so complex to be integrated into the NOS platform, due to its modular design. In fact, the introduction of the new module *CP-ABE Data Encryption* does not affect the behavior of the existing ones, as it emerges in Fig. 4; moreover, communications with the TA were already available in the previous version with sticky policies [5]. The main difficulty is represented by the management of the decryption keys in relation to the consumers subscribed to NOS.

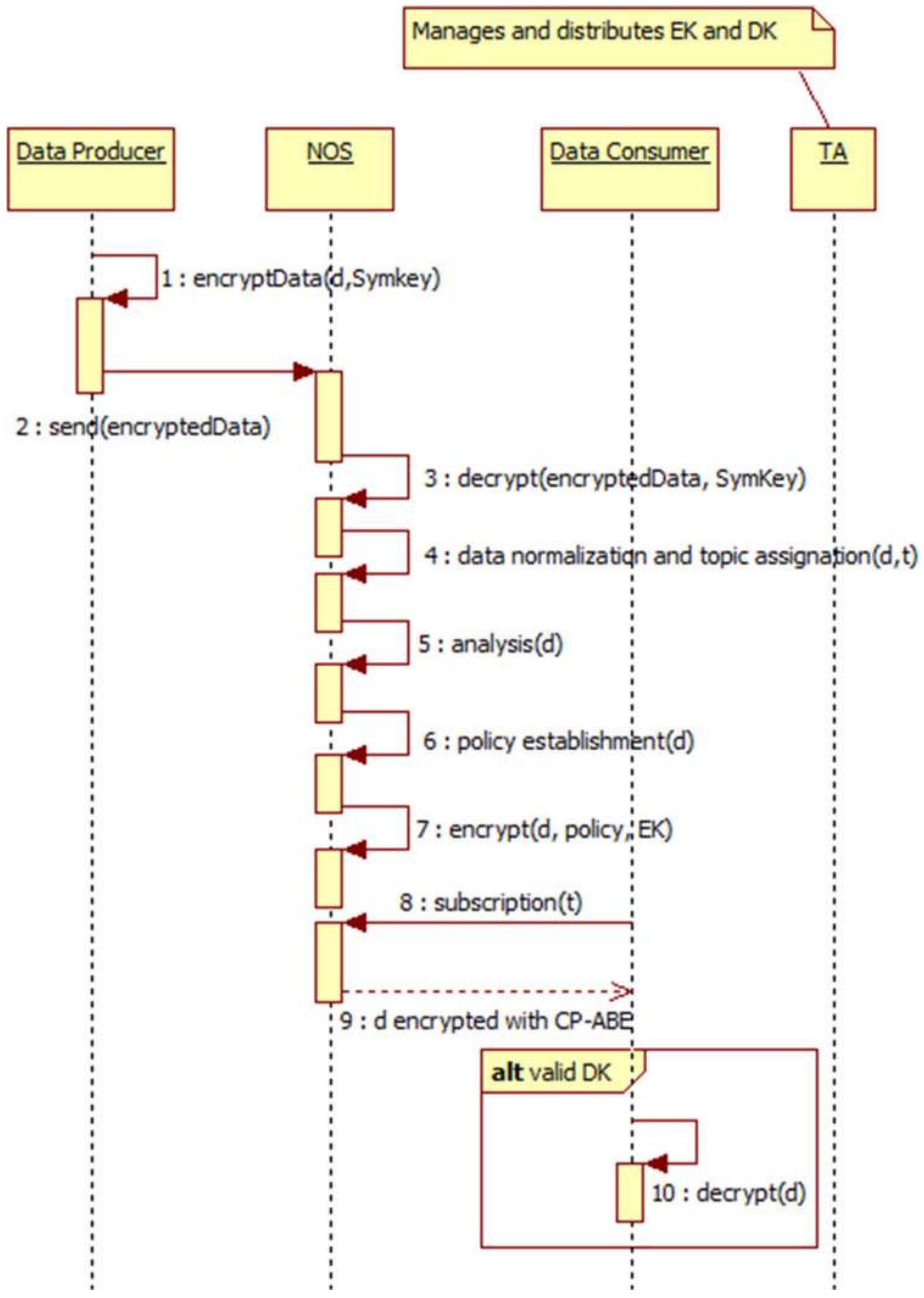
Manages and distributes EK and DK
Data Producer
NOS
Data Consumer
TA
1 : encryptData(d,Symkey)
2 : send(encryptedData)
3 : decrypt(encryptedData, SymKey)
4 : data normalization and topic assignation(d,t)
5 : analysis(d)
6 : policy establishment(d)
7 : encrypt(d, policy, EK)
8 : subscription(t)
9 : d encrypted with CP-ABE
alt valid DK
10 : decrypt(d)

Fig. 4 Scheme of CP-ABE-based data flow within the NOS system

The next section focuses on the key management mechanism, in order to clarify how decryption keys are distributed and assigned, and how decryption is enabled.

3.1 Key management

In a system adopting CP-ABE, the decryption keys must be distributed to the data consumers and revoked if they get compromised somehow. The mechanisms of distribution and revocation of decryption keys are often critical and complex. To implement such mechanisms, some additions to the basic Bethencourt's scheme have been introduced in NOS.

A version number is associated with each attribute, and inside the TA an *Attribute Version List* (AVL) is implemented, which is a list of all the attributes used in the system together with their latest version number. The AVL is created by the TA just after the execution of the Setup procedure, and all the version numbers are initialized to 1. From now on, an attribute is intended as including its version number. For example, the attribute "$attr$" will become "$attr\_vn$" where

n is the version number and "$attr\_v1$" and "$attr\_v2$" are considered two distinct attributes. Furthermore, the TA also detains a table of the unique identifiers of all the data consumers ($ConsID$) and their attribute sets, named *Consumers Attribute List* (CAL). This table is updated every time a new data consumer joins the system, by means of a subscription to a certain topic, since the IoT platform run on a publish&subscribed sharing, as described in Sect. 2.1.

It is supposed that each data consumer and the TA have their own pair of asymmetric keys (e.g., RSA or ECC) used for digital signature and encryption. Furthermore, it is assumed that the TA's public key is well known to all NOSs and data consumers, possibly obtained offline. Hence, the following procedures for key management are defined:

- *System initialization* In the *system initialization* procedure, the TA runs the CP-ABE primitive Setup, thus generating the couple (MK, EK). The TA has the responsibility to keep MK secret. Then, the TA creates the AVL by inserting all the attributes used in the system along with their version. Finally, the TA creates also the CAL, which is empty at the system initialization.
- *NOS join* The *NOS join* procedure is executed whenever a new NOS joins the system. The TA signs and communicates EK and the AVL to the NOS. The NOS can encrypt data using only attributes contained in the AVL.
- *Consumer join* The *consumer join* procedure is executed whenever a new data consumer joins the system. The data consumer requests a decryption key to the TA, declaring an attribute set γ that describes him/her. The TA has the responsibility to verify that the declared γ actually describes the consumer. Such attribute verification procedures are application-specific and fall outside the scope of the present paper. Every attribute inside γ must belong to the AVL maintained by the TA. Then, the TA executes the CP-ABE primitive KeyGen, generating a decryption key based on the previously mentioned γ. The TA updates the CAL by adding a tuple including the $ConsID$ of the consumer and its associated attribute set. Then, the TA signs the decryption key with its private key, and it encrypts the signed decryption key with the consumer's public key, which may have been acquired offline. The TA sends such signed and encrypted decryption key to the new consumer, which decrypts and authenticates it. If both operations are successful, the consumer accepts the decryption key and starts using it to decrypt data.
- *Data producer deployment* The *data producer deployment* procedure is executed whenever a new data producer is installed in the system. The data producer agrees with the associated NOS on a symmetric key, with which all the subsequent messages will be encrypted. Such a key agreement could be done in several ways, depending on the capabilities of the specific data producer. For example, NOS can transmit the symmetric key in clear to the data producer with a low-power wireless signal. This is a lightweight technique recommended by some IETF's RFCs [18] for smart home applications. It assumes that no eavesdropper is present at deployment time. If this assumption does not hold, more advanced key agreement protocols can be used, for example, anonymous or authenticated Diffie–Hellman.
- *Key revocation* The *key revocation* procedure is executed whenever a decryption key is compromised. This procedure drastically reduces the risk of data leakage, by invalidating and making useless the compromised decryption key. In order to ease the reading, the key revocation procedure is explained through an example in the following.

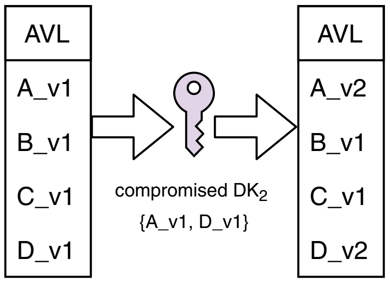

Fig. 5 Example of AVL update during a key revocation procedure. On the left, the AVL before the procedure of key revocation. In the middle, the key that has been compromised. On the right, the updated AVL

Suppose that the decryption key DK_2 of a consumer identified by *Cons2* has been compromised and must be revoked. The attribute set γ_2 associated with the decryption key includes the attributes A_v1, D_v1. To revoke DK_2, the TA updates the AVL by an increment in the version number of all these attributes, thus updating A_v1 to A_v2, and D_v1 to D_v2 (Fig. 5).

Then, the TA proceeds with re-generating the decryption keys of the *affected consumers* by executing the CP-ABE primitive KeyGen. The affected consumers are those consumers that have at least one attribute in common with the revoked decryption key. Let us suppose that the decryption key of the consumer identified by *Cons1* has one attribute (A_v1) in common with DK_2. Such a consumer is thus an affected one, and the TA re-generates his/her decryption key. The TA also updates the CAL table (see Fig. 6) by removing *Cons2*, whose decryption key has been revoked, and by upgrading A_v1 to A_v2 in the attribute set of the affected consumer *Cons1*.

Then, the TA proceeds to sign, encrypt (with the consumers public keys), and send the re-generated decryption key to each of the affected data consumer. Such an operation guarantees correct future decryption for affected data con-

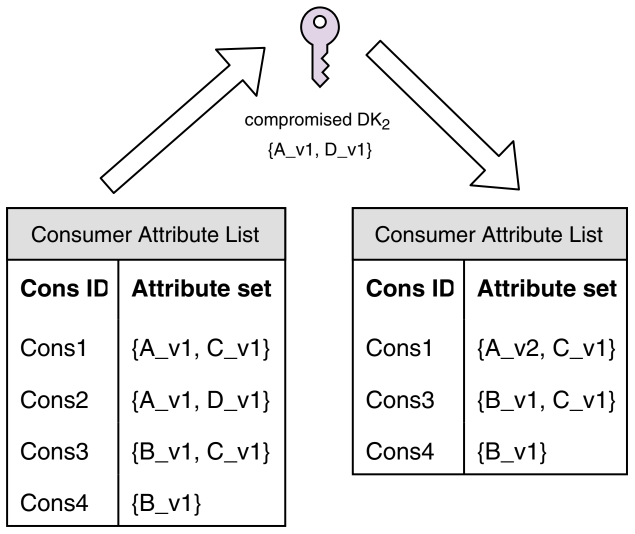

Fig. 6 Example of CAL update during a key revocation procedure. On the left, the table before the key revocation. On the right, the table after the key revocation

sumers, otherwise their old decryption keys will not decrypt new cipher-texts. Since the decryption key is encrypted, the TA can send it through an insecure channel (e.g., a simple email). Finally, the TA signs the updated AVL and sends it to the joined NOSs. From this moment, NOSs will encrypt with the new versions of the attributes. Such an operation makes useless the compromised decryption key, because the old version A_v1 and D_v1 are no longer used to encrypt data. To send the updated AVL, NOSs can do an MQTT subscription to the broker on a special-purpose topic dedicated to AVL updates from the TA, as just done in [19]. In this way, the TA can send a single AVL update to the broker, and the broker will eventually distribute it to all the NOSs.

For each key revocation, supposing n consumers and m NOSs in the system, the TA must send a single AVL update and $a \cdot n$ emails, where $a \in [0, 1]$ is the ratio of affected consumers. Such a ratio highly depends on the policy complexity. The authors in [20] computed that, in a large IoT system, the affected consumers of the average key revocation can be about $a = 12\%$ of the total consumers. Sending such quantity of emails should not be a problem with state-of-the-art bulk email software, given that key revocations should be rare events.

Note that while data encrypted *after* the key revocation procedure will not be decryptable by the revoked key, data encrypted *before* may still be accessible. This is because the CP-ABE scheme we employed in this paper provides no efficient re-encryption method [21], to transform a cipher-text labeled with an old-version attribute (e.g., A_v1) to another one labeled with a new-version attribute (A_v2). Of course, trivial (and inefficient) re-encryption methods are always possible, for example, by sending the old cipher-text to the TA to be re-encrypted with the new attribute version. We chose not to implement such methods to keep the system simple. However, this is surely an advantage point of the sticky policies approach, since it can protect old data without needing expensive re-encryption mechanisms.

4 Validation and experiments

For evaluating the approaches, just compared in Sect. 3, a threat model, an application scenario related to a smart home, and a test-bed for simulations are firstly presented. Then, numerical results are provided, with respect to the following metrics: storage occupancy, CPU load, and data retrieval delay.

4.1 Threat model and security analysis

We assume that each NOS has a copy of a trusted certification authority's public key. The TA owns a certificate released by such a CA. We then assume that each NOS knows the public key of the TA. which is used for digital signature, through the use of certificates.

The first threat considered is related to the violation attempts performed by malicious external parties. An external party is someone who acts from the outside of the IoT system, and he does not own any decryption key. His intent is to access encrypted information. In order to do so, he can try to eavesdrop a decryption key during a consumer join/subscription procedure or a key revocation procedure. Such an attack is avoided because, in both procedures, the decryption keys are encrypted with the consumer's public key. Alternatively, he can try to carry out an active *Man In the Middle* (MITM) attack. For CP-ABE, during the NOS join procedure, when the TA communicates the encryption key to the new joined NOS, the attacker can try to impersonate the TA and communicate to the NOS a malicious encryption key, so that he can decrypt all the cipher-texts produced by that NOS. Such an attack is avoided because the encryption key is digitally signed by the TA. Similar is the case of sticky policy paradigm, where, instead, the TA only communicates with NOSs, thus reducing the vulnerabilities. Hence, with respect to such a kind of attack, both the approaches (i.e., CP-ABE and sticky policies) are robust.

The second threat considers an external party that compromises a NOS. The effects of such an attack are many. First and foremost, the attacker has access to all the data that the smart objects will produce (and consequently send to the NOS) from that moment on. Past data encrypted with CP-ABE and stored in the NOS cannot be accessed by the attacker. However, the same cannot be said for past data encrypted with sticky policies and stored in the NOS, since it is symmetrically encrypted and thus the decryption key is known to the NOS. Secondly, the compromised NOS is able to manipulate the data it receives from the smart objects. Now we analyze the response of the system once the compromise has been solved, and the security hole that allowed it has been patched. In both approaches, the symmetric key used by each sensor managed by the NOS must be renewed, since they are also stored inside the NOS and they must be considered compromised. If the sticky policy approach is used, all the subscribers associated with the attacked NOS have to renew their credential. Since the subscriber credentials are stored in the NOS, they must be considered as compromised by the attacker. Instead, if the CP-ABE approach is used, no further cryptographic value has to be considered compromised. As a matter of fact, the NOS possesses only the encryption key, which is public, and therefore it is of no use for the attacker.

The third threat concerns possible colluding consumers wanting to acquire data that they cannot obtain singularly. Concerning this attack, the original Bethencourt's CP-ABE scheme [6] is natively collusion-resistant. This means that two or more consumers cannot combine their decryption keys in such a way to decrypt data that they cannot access singularly. Please refer to [6] for a mathematical proof of this.

To cope with the threats described above and, therefore, to resist active adversaries, the CP-ABE scheme must be indistinguishable under the adaptive chosen cipher-text attack (IND-CCA). Moreover, the signature scheme must be unforgeable under the chosen message attack (EUF-CMA). As the signature scheme, we chose the ECDSA algorithm which offers the needed security requirement. The original CP-ABE scheme that we employed (taken from the work of Bethencourt et al., [6]) is only proved to be indistinguishable under the chosen plaintext attack (IND-CPA). The proof of that is given by Bethencourt et al., and it is supported by the complexity of the bilinear Diffie–Hellman (BDH) problem. For being suitable against active adversaries, we converted the IND-CPA in an IND-CCA scheme, by applying the simple and efficient Fujisaki-Okamoto transformation [22], which only requires the random oracle model assumption. It is worth to note that, in this paper, the focus is on data security only. IoT devices and the IoT network can be attacked also on the control layer, for example, on the routing mechanisms. Secure IoT routing protocols [23] can help in this case, but they are outside the scope of the present paper.

4.2 Smart home scenario

An application scenario related to a typical smart home is used for conducting the performance evaluation, presented in Sect. 4.3. Data from real-world smart home test-bed have been gathered[1]; such data regard some smart meters installed in two houses, named A and B, which include, among the others, the electricity consumption related to: kitchen lights, bedroom lights, duct heater HRV, and HRV furnace. Note that the houses have a total of eight rooms and includes three full-time occupants. Measures are acquired by means of installed smart objects that collect electricity data every minute. Detailed information about such a smart home data set and on how information is thereby collected is available in [24].

Each person, which interacts with the houses, can be described by one or more of the following attributes:

- Landlord of the house X ($LanX$), who is the landlord of the house, but it does not imply that he/she lives there. The landlord might rent out the house. For example, the landlord can be a young man that has rent out an inherited apartment.
- Tenant of house X ($TenX$), who manages and lives in the house. The tenant has access to all the data generated in the house. The tenant and the landlord role may coincide. For example, a young woman that has recently bought an house and moved in, is both tenant and landlord of said house. Such an attribute is intended as a numerical, while it represents the date when the person was nominated tenant. In fact, a date can be represented as a numerical attribute equal to the number of days since a well-known date, as typical happens in computer science.
- Guest of the house X ($GueX$), who has access to the house, and he/she may not live there. For example, it can be an old couple's daughter that lives elsewhere, but she has Guest rights to check on her parents. He/she has access to a limited number of data. Such an attribute is intended as $TenX$, and it represents the date when the person was nominated guest.
- Expiring date for the tenant role of house X ($ExTenX$); it is also intended as a numerical attribute, as for $TenX$ and $GueX$, and it represents the date when the role of tenant will expire.
- Expiring date for the guest role of house X ($ExGueX$); it is also intended as a numerical attribute, as for $ExTenX$, and it represents the date when the role of guest will expire.

An example of attribute set γ for a person named Robert (R) is the following:

$$\begin{aligned}\gamma(R) = \{&LanB,\\ &TenA = 2/2/2000,\\ &ExTenA = 2/2/2020,\\ &TenB = 2/2/2015,\\ &ExTenB = 2/2/2020\}\end{aligned} \tag{5}$$

[1] http://traces.cs.umass.edu/index.php/Smart/Smart.

The above statement must be intended as follows: (i) Robert is the landlord of the house B; (ii) he is the tenant of house A since February 2nd 2000 and of the house B since February 2nd 2015; (iii) both his tenant roles will expire on February 2nd 2020. In such a scenario, versioning of attributes is not considered for readability.

Three possible data requests for each house are made available, even obtained from the above-mentioned data set:

– Access to the electrical data set: this is a data set related to the energy consumption of all the electronic and electric devices inside the house. Only the landlord and the tenants can access these data. To access them, a viable policy could be:

$$\mathcal{T}(\text{ElectricalDataset}) = \{LanX \vee (\text{today} \geq TenX \wedge \text{today} \leq ExTenX)\}, \tag{6}$$

where today is the date when data have been produced. Note that, due to the way in which numerical attributes are implemented in [6], the $\geq$ and $\leq$ operators return "false" in the case the numerical attribute does not exist in the decryption key.

– Video streaming: it provides live images from the inside of the house. Only who has actual access to the house can see video streaming from it. To request such a kind of data, the consumer must be an authorized as an tenant or a guest. Therefore, a viable policy could be:

$$\begin{aligned}\mathcal{T}(\text{VideoStream}) = \{(&\text{today} \geq TenX \wedge \\ &\text{today} \leq ExTenX) \vee \\ (&\text{today} \geq GueX \wedge \\ &\text{today} \leq ExGueX)\}.\end{aligned} \tag{7}$$

– Remote monitoring of house's current state: this implies the monitoring of relevant parameters such as temperature, humidity, lights switched on/off. Only the tenant can remotely monitor the status of the smart home. A viable policy could be:

$$\begin{aligned}\mathcal{T}(\text{Monitoring}) = \{(&\text{today} \geq TenX \wedge \\ &\text{today} \leq ExTenX)\}.\end{aligned} \tag{8}$$

The examples of policies just presented are derived from the attributes defined above. They will be used for the performance evaluation in Sect. 4.3.

4.3 Performance evaluation

In the experimental setup, NOS platform is deployed on a Raspberry Pi, which is a device widely used in IoT applications. The behavior of a set of consumers subscribing to obtain information about the smart homes (see Sect. 4.2) is emulated by means of a laptop, with the following features: (i) Core i7-4710HQ 2,5 GHz; (ii) 16 gigabytes of RAM; (iii) OS Ubuntu 16.04. The laptop uses WiFi IEEE 802.11 network to communicate with the Raspberry Pi. The same WiFi connection is also used for the communications with the MQTT broker and with the TA module, implemented as separate components, which interact with NOS on demand and run on separate laptops. A toolkit available online[2] has been used to implement the required CP-ABE primitives into the IoT system, as presented in Sect. 3.

Table 2 Configurations

Parameter	Value
Number of data producers	6
Number of data consumers	3
Number of attributes per policy	5
Data-rate provision from producers	1 pck/min
Requests' data-rate from consumers	1 request/min
Duration of the experiments	1 h
Time window of the data gathered from the data set	1 week

Sticky policy and CP-ABE approaches are compared w.r.t. the following metrics: storage and CPU load overhead, and data retrieval delay. The obtained results are compared on the basis of the application scenario, defined in Sect. 4.2. More in detail, 1 packet per minute is fetched from the simulated data sources and 1 data request per minute is simulated from the consumers. The number of data producers and consumers is set to 6 and 3, respectively; such values are derived from the simulation setups of two previous work on policy enforcement within the NOS architecture [17] [5], in order to ease the results' comparison and evaluation. Table 2 summarizes the setup parameters, while Fig. 7 sketches the interactions among the participants to the smart home scenario and the NOS platform. Note that bold text and arrows denote the interactions valid for CP-ABE, while the dashed arrow denotes the interactions valid for sticky policies; finally, thin arrows are common to both the approaches.

4.3.1 Storage, network, and CPU load

NOS components have the following storage requirements, which are different in the two approaches, as explained hereby:

– With regard to the approach based on sticky policies, the data sources and the consumers must store the credentials for ciphering the data to be transmitted to NOS. When

[2] http://acsc.cs.utexas.edu/cpabe/.

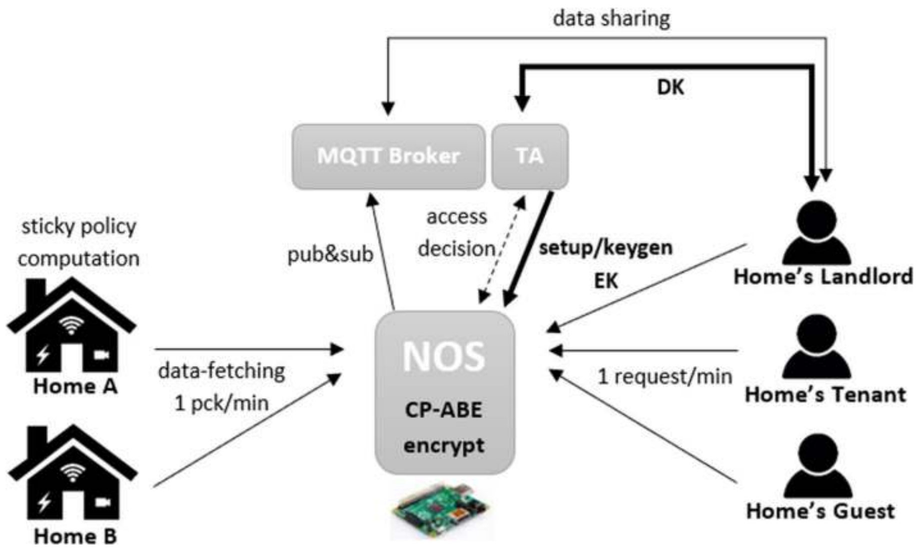

Fig. 7 Scheme of the performance evaluation setup

producers transmit data to NOS, they may also send the related sticky policy. Such an aspect unavoidably causes an increase of the traffic into the network, since not only the data are transmitted, but also the associated policy. An average increase of 0.5 kilobytes is measured for each transmitted data unit, considering the sticky policy format specified in [5], which approximately consists of 500 bytes. Whereas, adopting an approach based on CP-ABE, data sources have not to send a sticky policy along with the data to NOS, therefore such an increment is negligible; certainly, such an aspect represents a relevant advantage of adopting CP-ABE, because IoT networks usually transmit a huge amount of data. Note that in CP-ABE the packets' dimension increases once the encryption task has been performed by NOS.

– Starting for such premises, it is worth to note that the described behavior also influences the network load. In fact, for both the approaches, the information which is transmitted over the network are: (i) the data from producers; (ii) the consumers' requests; (iii) the consumers' responses (i.e., the data release). Figure 8 shows a reduced network load when adopting CP-ABE and it is mainly due to the fact that data sources do not transmit to NOS the data along with the policy (as happens for the sticky policy approach). The network load still remains lower for the CP-ABE approach with respect to the sticky policy one, even if there is an increment in the packet dimension when NOS performs the CP-ABE encryption.
– NOSs have to store different kinds of information. Yet, it is worth remarking that NOSs do not support persistent storage of IoT data for *Raw Data* and *Normalized Data* collections. In fact, incoming data are only temporarily cached on the NOSs' memory while being processed before being submitted to requesting consumers. Once data are further pushed to or pulled from the MQTT client (which handles the topics notification to subscribers), the data can be safely removed from NOSs. In both sticky policy and CP-ABE approaches, no further storage is required for policies, because the policies themselves are directly associated or embedded into the data. Hence, NOSs have not to store all the policies managed by the IoT system, as it happens in traditional approaches, as the one presented in [17]. However, it is fundamental to evaluate if it takes up more memory a sticky policy attached to the data or the same data encrypted with CP-ABE. The average memory occupancy on NOS at runtime is 10.2 megabytes with sticky policies, whereas with CP-ABE it slightly decreases to 8.4 megabytes. Note that such results have been obtained by equally setting the following factors for the two approaches: (i) the frequency of data fetching from sources (i.e., 1 packet/minute); (ii) the frequency of execution of the routines for removing data from non-persistent collections (in the actual environment, such a task is executed every 5 minutes); (iii) the number of sources (in the actual setup, 6 data producers are introduced). Obviously, for CP-ABE, the attributes' number highly influences the dimension of the encrypted data; however, it is true also for sticky policies. As a future work, a wider application context could be considered to perform a further assessment.
– Concerning the sticky policy based approach, the TA must store the whole set of the valid scopes and constraints used for sticky policies' composition [5]. The dimension of this storage obviously depends on the specific application domain. In the sample implementation, this was negligible. On the other hand, in the CP-ABE-based approach the TA has to maintain the AVL and CAL tables, whose sizes are also negligible as well in our sample implementation.

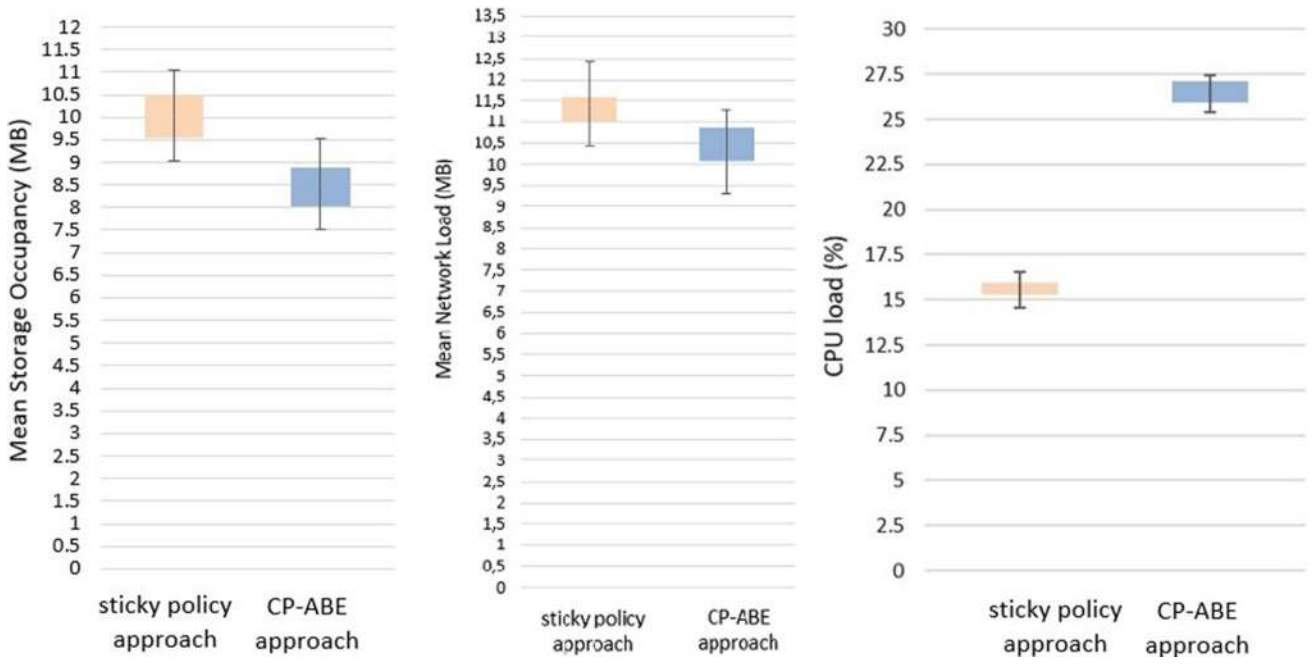

Fig. 8 Whiskers-box diagram of mean storage occupancy and CPU load comparison: sticky policies vs CP-ABE approach

The just presented analysis about memory occupancy reveals that adopting an approach based on CP-ABE would reduce the memory occupancy and the network load, thus improving the system's scalability in the presence of a higher amount of data. However, CP-ABE approach more affects the CPU load on NOS with respect to sticky policy approach, which, on the other side, increases the computational load on data sources. In fact, following the sticky policy approach, the data sources are in charge of computing the sticky policies and transmitting them along with the information to NOS, whereas, following the CP-ABE approach, the computational load is moved to NOS, which has to perform the encryption task on each incoming data. Hence, NOS shows a mean CPU load of 15.4% by adopting sticky policies, while, when running CP-ABE, the mean CPU load on NOS is 26.7%. Figure 8 sketches the comparison just discussed about storage occupancy, network and CPU load.

Taking into account such two perspectives, the most viable solution appears to be the one based on CP-ABE, because it brings the advantage of being more efficient for end-devices, since more powerful and secure devices, as NOSs, perform the heavier processing tasks. Such a point of view perfectly fits the principles of the emerging fog computing paradigm [25], which aims to: (i) reduce network's latency; (ii) prevent unnecessary network resources' consumption; (iii) enhance service availability; (iv) increase the robustness of the whole IoT system thanks to the removal of always-online points of failures into the security network infrastructure. Note that the TA is a single point of failure in the CP-ABE approach as well, but it has to be online only when revoking a decryption key, so it is hardly exposed to attacks.

4.3.2 Data retrieval delay

An important metric to be considered is the delay introduced by the enforcement framework using sticky policies with respect to CP-ABE. The main difference between the two mechanisms resides in how data are disclosed and, therefore, in how policies are evaluated. Note that "data retrieval delay" means the time from when a consumer requests topic subscription to when the same consumer receives and decrypts the relative data. Moreover, as emerged in Sect. 4.3.1, the packets transmitted by the data sources to NOS in case of sticky policy approach are approximately 0.5 kilobytes larger than the same packets sent with CP-ABE.

In the sticky policy approach, to obtain the access permission, the recipients can subscribe to certain topics and the subscription is accepted only if the request satisfies the requirements established by the sticky policies associated with the data. Access permissions are not locally evaluated by NOSs, but they are delegated to the TA; a query to the TA is sent for each occurring change and, in general, for each incoming request, thus clearly spending time for transmissions and processing. Different is the approach based on CP-ABE: once the subscribers obtained the decryption keys needed for disclosing the authorized information, they have no longer to make requests to the TA, which, as just said, can be offline most of the time.

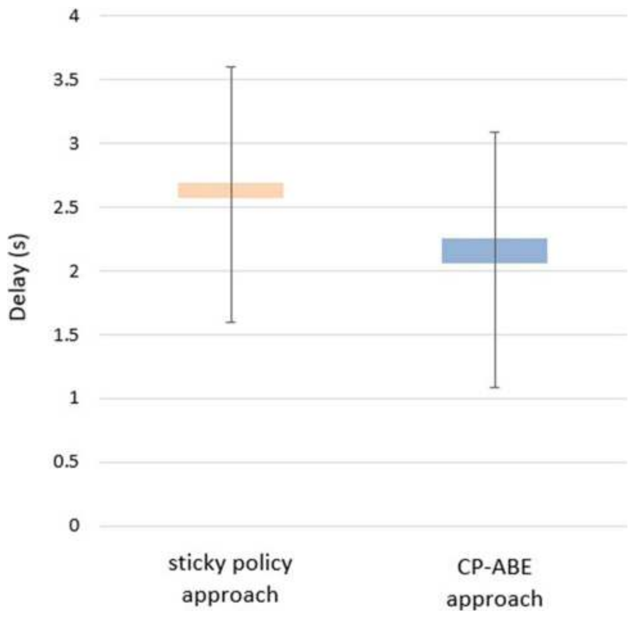

Fig. 9 Whiskers-box diagram of mean data retrieval delay comparison: sticky policies vs CP-ABE approach

For such a reason, the data retrieval delays are different, as shown in Fig. 9. Hence, CP-ABE allows to spend less time from two perspectives: (i) the data transmission from the source to NOS; (ii) the data disclosure. Figure 9 shows a comparison of the mean distribution of the delays generated by the two approaches, measured with the considered prototypical implementation over a period of one hour. Data rate strictly depends on the fetching of data acquisition of the used data set, which is every minute. The considered time window concerns a week of measurements.

Going in depth into the analysis of delays, Fig. 10 presents the encryption time required by CP-ABE for the three different kinds of data, managed within the smart home, which are: the electrical data set, the streaming video, and the remote monitoring, as explained in Sect. 4.2.

Finally, Fig. 11 shows the time required for decryption in CP-ABE, by varying the kind of data requested.

Summarizing, CP-ABE approach demonstrates to have several advantages, with respect to the adoption of sticky policies, in terms of memory occupancy on the IoT platform and delay.

5 Related work

Typically, current proposals, addressing security and privacy issues in the IoT, focus on data communications by enforcing data exchanges according to strict protection constraints, considering, at the same time, the heterogeneity of devices and communication technologies. In fact, devices can be characterized by different protocols. For example,

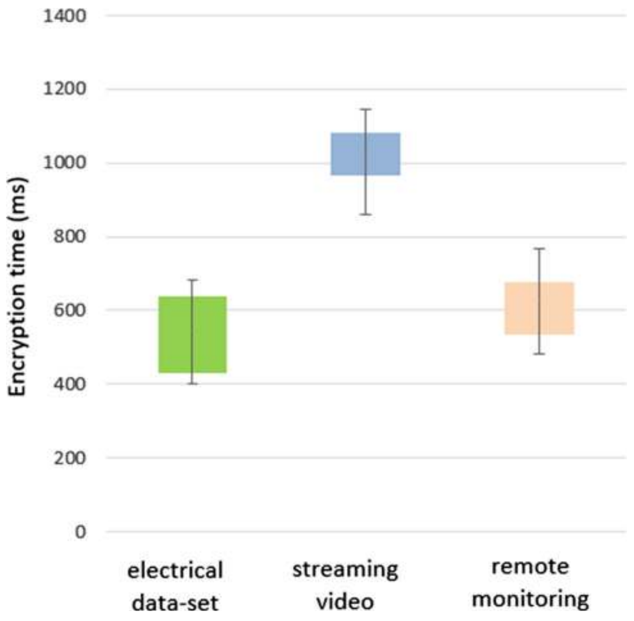

Fig. 10 Whiskers-box diagram of mean encryption time required by CP-ABE

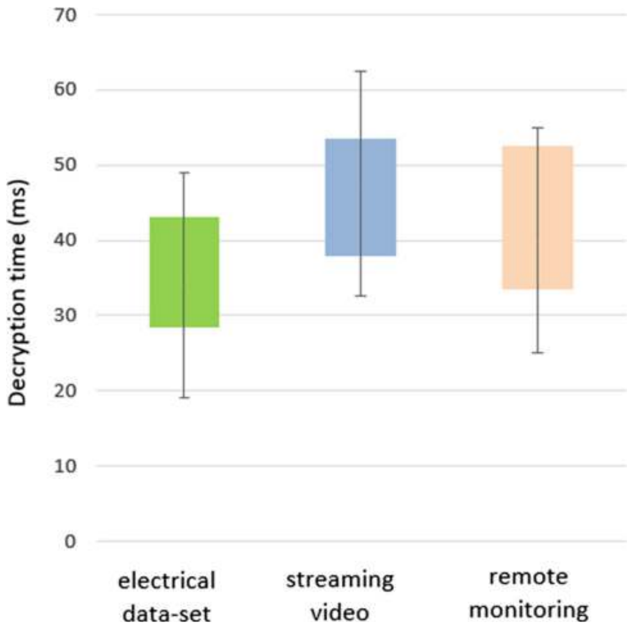

Fig. 11 Whiskers-box diagram of mean decryption time required by CP-ABE

many smart devices can natively support IPv6 communications [26] [27], while other existing deployments might not support the IP protocol within the local area scope and this requires the design of ad-hoc gateways and middleware [28]. This is the reason for introducing NOS middleware in the envisioned solution.

Relevant contributions on security-oriented IoT middleware include: VIRTUS [29], which relies on the open eXtensible Messaging and Presence Protocol (XMPP) to provide secure event-driven communications; Otsopack [30] and

Naming, Addressing and Profile Server (NAPS) [31], which are data-centric frameworks based on the usage of HTA and REpresentational State Transfer (REST) interfaces. With respect to such frameworks, NOS is more recent and adopts a lightweight technology, based on Node.js in an event-driven fashion, which perfectly fits the requirements of IoT applications. Also various projects have the final purpose of delivering a framework able to dynamically integrate user data (e.g., location, behavior) in privacy and security protocols, as reported in [32]. As an example, within the EU FP7 project, the RERUM middleware is based on the open-source OpenIoT that was selected as the most efficient solution, mainly due to its open-source nature and the fact that it was developed under the concepts of the broadly accepted Architectural Reference Model (ARM) of the EU lighthouse project Internet of Things Architecture (IoT-A) [33]. RERUM added a service manager and a security server, acting all security- and privacy-related functionalities; such a server can be developed as a standalone component talking to the RERUM middleware via pre-defined interfaces or as an integrated component of the RERUM middleware. Then, the communications among the RERUM middleware and the IoT gateways take place though a virtual private network (VPN), in order to ensure that only authorized gateways are sending data to the RERUM middleware in a secure way. An access control mechanism, integrated with the RERUM middleware, is also in place. As previously stated, NOS middleware is adopted in this work due to its lightweight nature, which differs from the complexity of the solution proposed by RERUM project. Other solutions make also use of cloud computing [34], but the role of cloud is out of the scope of the present work.

The IoT middleware named NOS, firstly implemented and presented in [4], tried to definitely fill the gap by providing an efficient processing and assessment of the IoT data. Such functionalities have been further coupled with a policy enforcement framework based on sticky policies [5], and relevant security requirements have been addressed, as detailed in Sect. 2.1. The novelty introduced in this paper is the ABE paradigm's integration into NOS and its comparison with the sticky policy based approach. It is worth to remark that the presented solutions based on sticky policies and CP-ABE mechanisms are both conceived to include the presence of multiple NOSs. They can easily and securely share data, acting as intermediaries with each other. A mechanism for policies' synchronization could be required by a specific application domain. With this regard, a solution able to synchronize the policies among different NOSs has been already provided in [19].

Regarding ABE, some IoT-focused cryptographic schemes [35–37] and architectures [20,38–41] can be found in the literature. In [35], the first KP-ABE scheme for wireless sensor networks (WSNs) is presented. The proposed scheme is composed by one trusted network controller, several users, and several sensor nodes. Each user owns a secret key generated by the network controller, according to a policy that describes the type of data he/she can access. Each sensor node is pre-loaded with a set of attributes and their relative public quantities, generated by the network controller. In [36], a lightweight KP-ABE scheme for the IoT is presented. The math behind the proposed scheme is based on elliptic-curve cryptography, rather than pairing-based cryptography as the majority of the other ABE schemes. This makes the scheme more efficient from the point of view of encryption and decryption times. In [37], a CP-ABE scheme allowing for constant-size keys and cipher-texts is presented. This makes the scheme more scalable, especially in case of battery-limited devices and bit-rate-limited channels, as in the typical IoT application. The scheme allows only AND operators to be used in the Boolean formulas of the policies, so it provides for limited expressiveness. The above cryptographic schemes are unsuitable to be used in the present paper, which aims at comparing CP-ABE and sticky policy approaches. This is either because they follow a KP-ABE approach instead of a CP-ABE one ([35,36]), or because they provide for too little policy expressiveness compared to sticky policies ([37]).

In [38], a secure publish-subscribe protocol for medical wireless body area network (WBANs) using ABE is proposed. The conceived architecture follows a star-topology network, where a smart phone (or a similar device) manages the communication among various nodes placed over/inside the user's body, monitoring his/her health conditions. Each node can publish its data and subscribe to data generated from other nodes. In [39], a secure MQTT for IoT is introduced, along with the possibility of using ABE. The proposed architecture is composed by one public key generator (PKG), one broker and several devices, which can act both as subscribers and publishers. Each device owns the public key and a secret key associated with some attributes that describes its features. Then, each device subscribes to certain topics in order to receive the data of interest. In [20], a system for smart cities using ABE is presented. The application offers a service of real-time road monitoring in a smart city scenario. Smart objects (e.g., cameras) are placed along the roads and store their sensed data on a cloud storage service. Users can pay a subscription and obtain an ABE decryption key, in order to retrieve and decrypt the video streams of the city traffic in real time. In [40], a system for protecting location data in smart buildings using CP-ABE is presented. Their approach is based on the concept of "bubbles," which are coalitions of smart objects defined according to relationships between their owners. As emerged, ABE schemes are not widely adopted in IoT scenarios yet. For such a reason, the analysis conducted in such a paper contributes in assessing ABE capabilities, feasibility, and potentialities within an IoT middleware in an IoT typical scenario.

6 Discussion and conclusions

The paper has presented a comparison between CP-ABE and sticky policy approaches in a smart home environment. The analysis, conducted by means of a prototypical implementation of the two solutions, revealed the potentialities of CP-ABE in guaranteeing a secure-aware and efficient data flow management with respect to approaches based on sticky policies. In fact, with CP-ABE, the dimension of the packets sent from the data sources to the IoT platform is reduced; also, the memory occupancy on the IoT platform itself is lower than the one obtained by adopting the sticky policies approach. Furthermore, the mechanisms provided by means of CP-ABE limits the delay of packets' transmission from the producer to the consumer. One main drawback of CP-ABE is the CPU load required for performing the encryption task. Concerning robustness toward different possible attacks, CP-ABE appears to be more resilient. Based on the analysis performed throughout the paper and based on the results obtained in Sect. 4.3, we suggest some scenarios in which CP-ABE is recommended over sticky policy and vice versa, to help developers in choosing the right technique based on their needs. The experimental results show us that the sticky policy approach strains less the CPU compared to the CP-ABE approach, therefore suggesting us that a NOS is able to manage more smart objects when the sticky policy approach is used. Due to this, the sticky policy approach is recommended if the developers want to maximize the amount of smart objects managed by a single NOS. Furthermore, after a revocation happens, data stored on NOSs are readily encrypted under new credentials, a feature that is not available when using CP-ABE. These characteristics make sticky policy ideal in environments with an high density of smart objects, like a smart building, and/or for applications that needs secrecy of past data. Instead, the CP-ABE approach leverages the fact that usually the NOSs are notably more resourceful than the IoT smart objects, and it offers limited interactions with the TA. These characteristics make CP-ABE ideal in environments with a low density of smart objects, in which the NOS(s) can easily manage the encryption of the data generated by smart objects. The CP-ABE approach suits both small-scale scenarios and wide-area low-density scenarios. Small-scale scenarios are, for example, the smart home presented in this work, or a small factory like the one presented in [42]. Wide-area low-density scenarios are, for example, a smart city with many smart objects (e.g., cameras, smart street-light) scattered throughout the city, in which each NOS can manage a small cluster of them. This is because CP-ABE is inherently more scalable than sticky policies w.r.t. the TA interactions, as just said. If the sticky policy approach is used in the last scenario, the TA could have been a bottleneck. With regard to the future work, we plan to evaluate the presented solution in presence of more than one NOS and considering the TA as a fog- or cloud-based solution, in order to prevent it from representing a single point of failure and a bottleneck for the IoT system. Another goal is trying to analyze the correlation among the number of NOSs and data sources/users to manage. More in detail, the authors would investigate how many NOSs are required to efficiently manage a certain distribution of IoT entities. Moreover, the mechanism of key revocation will be deeper studied, evaluating the time and the cost required to secure the IoT system in case of keys' or policies' revocation. Finally, power consumption on real IoT devices will be investigated.

Funding Open access funding provided by Universitá degli Studi dell'Insubria within the CRUI-CARE Agreement.

Compliance with ethical standards

Conflict of interest Sabrina Sicari declares that she has no conflict of interest. Alessandra Rizzardi declares that she has no conflict of interest. Michele La Manna declares that he has no conflict of interest. Pericle Perazzo declares that he has no conflict of interest. Gianluca Dini declares that he has no conflict of interest. Alberto Coen-Porisini declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
2. Sahai, A., Waters, B.: Fuzzy identity-based encryption. Eurocrypt **3494**, 457–473 (2005)
3. Pearson, S., Mont, M.C.: Sticky policies: an approach for managing privacy across multiple parties. Computer **44**(9), 60–68 (2011)
4. Sicari, S., Rizzardi, A., Miorandi, D., Cappiello, C., Coen-Porisini, A.: A secure and quality-aware prototypical architecture for the internet of things. Inf. Syst. **58**, 43–55 (2016)
5. Sicari, S., Rizzardi, A., Miorandi, D., Coen-Porisini, A.: Security towards the edge: sticky policy enforcement for networked smart objects. Inf. Syst. **71**, 78–89 (2017)
6. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, 2007. SP'07. pp. 321–334 (2007)

7. (1999) IBM and eurotech, ”mqtt v3.1 protocol specification”. http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html
8. Rizzardi, A., Sicari, S., Miorandi, D., Coen-Porisini, A.: AUPS: an open source AUthenticated publish/subscribe system for the internet of things. Inf. Syst. **62**, 29–41 (2016)
9. Node.JS (2009). http://nodejs.org/
10. MongoDB. (2009). http://www.mongodb.org/
11. Mosquitto ”an open source mqtt v3.1/v3.1.1 broker”. (2009). http://mosquitto.org
12. Karjoth, G., Schunter, M., Waidner, M.: Privacy-enabled services for enterprises. In: 13th International Workshop on Database and Expert Systems Applications, 2002. Proceedings, IEEE, pp. 483–487 (2002)
13. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and Communications Security, pp. 89–98 (2006)
14. Ambrosin, M., Anzanpour, A., Conti, M., Dargahi, T., Moosavi, S.R., Rahmani, A.M., Liljeberg, P.: On the feasibility of attribute-based encryption on Internet of Things devices. IEEE Micro **36**(6), 25–35 (2016)
15. Girgenti, B., Perazzo, P., Vallati, C., Righetti, F., Dini, G., Anastasi, G.: On the feasibility of attribute-based encryption on constrained IoT devices for smart systems. In: 2019 IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, pp. 225–232 (2019)
16. Ambrosin, M., Conti, M., Dargahi, T.: On the feasibility of attribute-based encryption on smartphone devices. In: Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems, ACM, pp. 49–54 (2015)
17. Sicari, S., Rizzardi, A., Miorandi, D., Cappiello, C., Coen-Porisini, A.: Security policy enforcement for networked smart objects. Comput. Netw. **108**, 133–147 (2016)
18. Baccelli, E., Cragie, R., Der Stok, P., Brandt, A.: Applicability Statement: The Use of the Routing Protocol for Low-Power and Lossy Networks (RPL) Protocol Suite in Home Automation and Building Control. RFC 7733, RFC Editor, (2016). https://www.rfc-editor.org/rfc/rfc7733.txt
19. Sicari, S., Rizzardi, A., Miorandi, D., Coen-Porisini, A.: Dynamic policies in internet of things: enforcement and synchronization. IEEE Internet Things J. **4**, 2228–2238 (2017)
20. Rasori, M., Perazzo, P., Dini, G.: ABE-Cities: an attribute-based encryption system for smart cities. In: Proceedings of IEEE SMARTCOMP 2018 (to appear), pp. 1–8 (2018)
21. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Infocom, 2010 Proceedings IEEE, pp. 1–9 (2010)
22. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual International Cryptology Conference, Springer, pp. 537–554 (1999)
23. Perazzo, P., Vallati, C., Arena, A., Anastasi, G., Dini, G.: An implementation and evaluation of the security features of RPL. In: International Conference on Ad-Hoc Networks and Wireless, Springer, pp. 63–76 (2017)
24. Barker, S., Mishra, A., Irwin, D., Cecchet, E., Shenoy, P., Albrecht, J.: Smart*: an open data set and tools for enabling research in sustainable homes. SustKDD **111**, 112 (2012)
25. Yi, S., Li, C., Li, Q.: A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data, ACM, pp. 37–42 (2015)
26. Palattella, M., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L., Boggia, G., Dohler, M.: Standardized protocol stack for the internet of (important) things. Commun. Surv. Tutor. IEEE **15**(3), 1389–1406 (2013)
27. Bagci, I., Raza, S., Chung, T., Roedig, U., Voigt, T.: Combined secure storage and communication for the Internet of Things. In: 2013 IEEE International Conference on Sensing, Communications and Networking, SECON 2013, New Orleans, LA, United States, pp. 523–631 (2013)
28. Boswarthick, D., Elloumi, O., Hersent, O.: M2M Communications: A Systems Approach, 1st edn. Wiley, Hoboken (2012)
29. Conzon, D., Bolognesi, T., Brizzi, P., Lotito, A., Tomasi, R., Spirito, M.: The VIRTUS middleware: an XMPP based architecture for secure IoT communications. In: 2012 21st International Conference on Computer Communications and Networks, ICCCN 2012, Munich, Germany, pp. 1–6 (2012)
30. Gòmez-Goiri, A., Orduna, P., Diego, J., de Ipina, D.L.: Otsopack: lightweight semantic framework for interoperable ambient intelligence applications. Comput. Hum. Behav. **30**, 460–467 (2014)
31. Liu, C.H., Yang, B., Liu, T.: Efficient naming, addressing and profile services in Internet-of-Things sensory environments. Ad Hoc Netw. **18**, 85–101 (2013)
32. Sicari, S., Rizzardi, A., Grieco, L.A., Coen-Porisini, A.: Security, privacy and trust in internet of things: the road ahead. Comput. Netw. **76**, 146–164 (2015)
33. Moldovan, G., Tragos, E.Z., Fragkiadakis, A., Pohls, H.C., Calvo, D.: An IoT middleware for enhanced security and privacy: the RERUM approach. In: 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), IEEE, pp. 1–5 (2016)
34. Mukherjee, B., Wang, S., Lu, W., Neupane, R., Dunn, D., Ren, Y., Su, Q., Calyam, P.: Flexible IoT security middleware for end-to-end cloud-fog communication. Future Gener. Comput. Syst. **87**, 688–703 (2018)
35. Yu, S., Ren, K., Lou, W.: FDAC: toward fine-grained distributed data access control in wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. **22**(4), 673–686 (2011)
36. Yao, X., Chen, Z., Tian, Y.: A lightweight attribute-based encryption scheme for the Internet of Things. Future Gener. Comput. Syst. **49**, 104–112 (2015). https://doi.org/10.1016/j.future.2014.10.010
37. Odelu, V., Das, A.K., Khan, M.K., Choo, K.K.R., Jo, M.: Expressive CP-ABE scheme for mobile devices in IoT satisfying constant-size keys and ciphertexts. IEEE Access **5**, 3273–3283 (2017)
38. Picazo-Sanchez, P., Tapiador, J.E., Peris-Lopez, P., Suarez-Tangil, G.: Secure publish-subscribe protocols for heterogeneous medical wireless body area networks. Sensors **14**(12), 22619–22642 (2014)
39. Singh, M., Rajan, M., Shivraj, V., Balamuralidhar, P.: Secure MQTT for Internet of Things (IoT). In: 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT), IEEE, pp. 746–751 (2015)
40. Hernández-Ramos, J.L., Pérez, S., Hennebert, C., Bernabé, J.B., Denis, B., Macabies, A., Skarmeta, A.F.: Protecting personal data in IoT platform scenarios through encryption-based selective disclosure. Comput. Commun. **130**, 20–37 (2018)
41. Rasori, M., Perazzo, P., Dini, G.: A lightweight and scalable attribute-based encryption system for smart cities. Comput. Commun. **149**, 78–89 (2020)
42. La Manna, M., Perazzo, P., Rasori, M., Dini, G.: Fabelous: an attribute-based scheme for industrial internet of things. In: 2019 IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, pp. 33–38 (2019)