A secure and quality-aware prototypical architecture for the Internet of Things

Sabrina Sicari[*‡], Alessandra Rizzardi[*], Daniele Miorandi[§], Cinzia Cappiello[†], Alberto Coen-Porisini[*]
[*]Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria,
via Mazzini 5 - 21100 Varese (Italy)
[§]U-Hopper, via A. da Trento 8/2, 38122 Trento, Italy
[†]Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
[‡]Corresponding author
Email: {sabrina.sicari; alessandra.rizzardi; alberto.coenporisini}@uninsubria.it,
daniele.miorandi@u-hopper.com, cinzia.cappiello@polimi.it

***Abstract*—The increasing diffusion of services enabled by Internet of Things (IoT) technologies raises several risks associated to security and data quality. Together with the high number of heterogeneous interconnected devices, this creates scalability issues, thereby calling for a flexible middleware platform able to deal with both security threats and data quality issues in a dynamic IoT environment. In this paper a lightweight and cross-domain prototype of a distributed architecture for IoT is presented, providing minimum data caching functionality and in-memory data processing. A number of supporting algorithms for the assessment of data quality and security are presented and discussed. In the presented system, users can request services on the basis of a publish/subscribe mechanism, data from IoT devices being filtered according to users requirements in terms of security and quality. The prototype is validated in an experimental setting characterized by the usage of real-time open data feeds presenting different levels of reliability, quality and security.**

I. Introduction

The Internet of Things (IoT) revolution is turning everyday objects into *smart* ones. Such *smart things* are able to interact among themselves and with the environment they are in in order to fulfill a given goal [1]. As a result, a global network infrastructure is being created, supporting the provisioning of innovative and customized services to individuals and businesses in different application domains. The resulting system may include an extremely large number of heterogeneous devices, raising integration and scalability challenges to be addressed.

Security & privacy are widely acknowledged to represent critical issues in such a context [2]. On the one hand, the confidentiality and the integrity of the transmitted and stored information has to be guaranteed, and authentication and authorization mechanisms have to be provided to prevent unauthorized users or devices to improperly access the system. On the other hand, privacy of users, understood as ability to support data protection and users anonymity has to be ensured, a critical aspect in particular in the presence of personal and/or sensitive information [3]. Beyond security, also data quality represents an essential requirement for the adoption at scale of IoT services. The provided information should be accurate, timely and complete, since, in some scenarios, errors or missing values might have critical impact on actions or decisions of the IoT system itself [4]. Indeed, as in IoT-enabled services and applications may make use of different data sources, the user (or the application itself) has to be aware of the security and quality level of the data being accessed, in order to take informed decisions about their usage.

As a result, what emerges is the need of a system able to deal with heterogeneous data sources and to evaluate the security and the quality of the information being collected, processed and transmitted, possibly in real-time and in an automatic manner. Furthermore, such a system shall be able to work in the absence of a priori complete knowledge of the sources themselves, since IoT environments are highly dynamic and different kinds of attack may occur. In fact, in such a scenario, data may be compromised by rogue devices, hindering the correctness and confidentiality of the information (e.g., data integrity violation, man-in-the-middle attacks, packet sniffing). Source authentication issues (e.g., compromised keys, session violation) shall also be accounted for. In order to deal with such issues, a mechanism for the assessment of data quality and security is proposed in this paper, supported by a number of novel algorithms aimed at analyzing the data sources as well as the data they generate over time. As far as security assessment is concerned, the presented algorithm is new and, according to the authors' knowledge, the first of its kind. Its aim is to assign a level of robustness to each data source according the following security features: integrity, confidentiality, authentication system, privacy. Therefore, the proposed solution does not tackle the security attacks directly, but aims at minimizing the associated risks by letting users and applications be aware of the security and data quality level.

The proposed solution is integrated in an existing IoT middleware, named *NetwOrked Smart objects (NOS)* [5]. NOS are conceived as computationally powerful devices connected to create a distributed processing and storage layer able to process the data acquired from large-scale IoT deployments close to the actual data sources. NOSs collect the data generated by nearby IoT devices, process them and finally transmit the processed data on a publish/subscribe broker. Such a middleware includes provisionings for users and applications to dynamically specify the level of security and data quality suitable for their own purpose. The distributed architecture automates the deployment of adequate filters for ensuring that only qualified data is being used by the actual service. This represents a clear innovation over conventional one-size-fits-all approaches, which provide the same information to all consumers, often without considering his/her requirements in terms pf security, privacy and data quality.

The algorithms integrated within the prototype are validated in an experimental setup characterised by the usage of real-time open data feeds and, contextually, assessing the resulting security and quality level. With respect to the original highly modular and lightweight prototype presented in [5], NOS functionalities are significantly improved in this work, enhancements including: (i) a set of methods for the distributed and autonomic management and run-time optimization of the middleware platform; (ii) a set of secure and privacy-aware mechanisms and their integration in the proposed platform; (iii) a set of data quality assessment methods that can be used for different types of data and in different scenarios; (iv) standardized interfaces and data models for applications/services to access qualified IoT information following a publish/subscribe architecture, where raw data are enriched with metadata specifying their security and quality levels.

The paper is organized as follows. Section II reviews the relevant literature and state of the art. Section III presents the system architecture and explains the storage and the data management aspects, as well as all the operating modules and their functions. Section IV and V present the prototype and its validation scenario, in order to demonstrate its effectiveness in a real-world IoT context. Section VI concludes the paper providing directions for future research.

II. Related Work

One of the main factors limiting the growth and take-up of IoT is the lack of a set of standardised tools, platforms and interfaces able to provide interoperability across different vendors of hardware and software solutions as well as across diverse vertical domains.

In the last few years, many initiatives tried to bridge this gap, reusing concepts, techniques and protocols from the Internet domain. For example, in recent years, the widespread adoption of web services has provided a standard framework to enable systems' interoperability according to the principles of Service Oriented Architectures (SOA). Service-oriented Communications (SOC) technologies manage web services by creating a virtual network and adapting applications to the specific needs of users rather than users being forced to adapt to the available functionality of applications [6],[7]. Although the trend towards the adoption of SOA architectural principles in the IoT domain is shared by the majority of the scientific community, at the moment the state of the art in this area is still somehow limited [8],[9].

Due to the very large number of heterogeneous technologies being used in IoT systems, several middleware layers have been proposed to enforce the integration and the security of devices and data within the same information network. Typically, they enforce data to be exchanged according to strict protection constraints. Heterogeneity of devices and communication technologies in IoT has to be accounted in the design of such middleware architecture. Indeed, while many smart devices can natively support IPv6 communications [10] [11], existing deployments might not support the IP protocol within the local area scope, thus requiring ad hoc gateways and middlewares [12]. Recent works on IoT middlewares include: VIRTUS [13], which relies on the open eXtensible Messaging and Presence Protocol (XMPP) to provide secure event-driven communications; Otsopack [14] and Naming, Addressing and Profile Server (NAPS) [15], which are data-centric frameworks based on the usage of HTTP and REpresentational State Transfer (REST) interfaces.

An attempt to provide a lightweight and flexible middleware for IoT applications is at the heart of the work reported in [5] where, starting from a general UML conceptual model and a high level design of IoT architectures [4], [16], a prototypical implementation of an IoT middleware is presented. Note that the solution proposed in [5] tries not only to manage information provided by heterogeneous sources, but also to evaluate the data provided by the IoT system both in terms of security and quality.

What emerges from the analysis of the relevant literature and state of the art, presented in this section, is that no available middleware approach is actually able to address comprehensively security, privacy and data quality issues in a IoT context. The approach proposed in [5] represents a first attempt in this direction, yet limited to a deterministic assessment of the security level. Instead, in this paper, an automatic reasoning mechanism for the assessment of the security and quality of data is added to the existing prototypical implementation, thus improving the actual solution with an innovative and, to the best of authors' knowledge, new algorithm suitable for the IoT context.

Besides the scientific literature, it is worth in this context to mention some relevant EU projects, such as FP7 COMPOSE [17], iCORE [18], IoT.EST [19], Ebbits [20], uTRUSTit [21] and Butler [22]. Only the last two focus on security aspects.

The FP7 COMPOSE (Collaborative Open Market to Place Objects at your Service) project [17] aims to design and develop an open marketplace for IoT data and services. The basic concept underpinning such an approach is to treat smart objects as services, which can be managed using standard service-oriented computing approaches and can be dynamically composed to provide value-added applications to end users.

The iCORE project (iCORE) [18] aims to empower the IoT with cognitive technologies and is focused around the concept of virtual objects (VOs). VOs are semantically enriched virtual representation of the capabilities/resources provided by real-world objects. Through the inception of VOs it becomes possible to easily re-use Internet-connected objects through different applications/services, also supporting their aggregation into more complex artifacts (composite virtual objects - CVOs). VOs provide a unified representation for smart objects, hiding from the application/service developers low-level technological details as well as any underlying technological heterogeneity. VOs also provide a standardised way to access objects features and resources. One key element in the iCORE project is the use of advanced cognitive techniques for the management and composition of VOs in order to improve IoT applications and better match user/stakeholder requirements. Application scenarios considered include ambient assisted living, smart office, transportation and supply chain management.

A dynamic architecture for services orchestration and self–adaptation has been proposed in IoT.EST (Internet of Things Environment for Service Creation and Testing) [19]. The project defines a dynamic service creation environment that gathers and exploits data from sensors and actuators making use of different communication technologies and formats. Such an architecture deals with issues such as the composition of business services based on re-usable IoT service components, the automated configuration and testing of services for "things" and the abstraction of the heterogeneity of underlying technologies to ensure interoperability.

The Ebbits project [20] designed a SOA platform based on open protocols and middleware, effectively transforming IoT subsystems or devices into web services with semantic resolution. The goal was to allow businesses to integrate IoT into mainstream enterprise systems and to support interoperable end-to-end business applications.

Finally, dealing with security, privacy and trust issue there are the uTRUSTit [21] and Butler [22] projects. The approach pursued in the former one is to integrate the user directly in the trust chain, guaranteeing transparency in the underlying IoT security and reliability properties. If successful, uTRUSTit approach shall enable system manufacturers and system integrators to express the underlying security concepts to users in a comprehensible way, allowing them to reason on the trustworthiness of such systems. Butler aims to allow users to manage their distributed profile by allowing data duplication and identity control over different applications. The final purpose is to deliver a framework able to dynamically integrate user data (e.g., location, behaviour) in privacy and security protocols.

As far as data quality is concerned, several literature contributions recognize it as one instrumental issue in IoT research. In [23], authors claim the need of control over data sources to ensure their validity, information accuracy and credibility. Data accuracy is also addressed in [24]; the authors observe that the presence of many data sources raises the need to understand the quality of such data. In particular, they state that the data quality dimensions to consider are accuracy, timeliness and the trustworthiness of the data provider. Anomaly detection techniques are widely employed in various scenarios to remove noise and inaccurate data in order to improve data quality. The huge number of data sources in IoT is considered a positive aspect for data fusion and for the extraction and provisioning of advanced services. Besides temporal aspects (i.e., currency) and data validity, a related work adds another important dimension such as availability [25]. The authors of such work define new metrics for the aforementioned quality dimensions in the IoT context and evaluate the quality of the real-world data available on the open IoT platform Cosm. In particular, they show that data quality problems are frequent and they should be adequately tackled or, at least, users should be aware of the poor quality of the used data sources. In fact, IoT systems will be part of everyday life in the same way mobile phones have become an integrated part of our life. Trustworthiness, security and privacy implications of IoT are vast, and must be made accessible to users.

III. Architecture

Two main entities can be identified in an IoT context: (i) the nodes, intended as data sources and represented by heterogeneous devices (e.g., wireless sensor networks, RFID, NFC, actuators, social networks); (ii) the users, who access services making use of IoT data through Internet-connected mobile devices (e.g., smartphone, tablet). In our architecture [16], represented in Fig. 1, we also have a layer (the NOS layer) in charge of processing and managing the data generated by the nodes. NOSs are networked smart nodes deployed in a distributed manner and do not present strict constraints in terms of resources and computational capabilities. They are able to interface with nearby IoT devices and to process the acquired data closer to the sources than a centralized platform. In the next sections, through a bottom-up analysis, the components of a NOS are detailed. The description includes (i) the southbound interfaces (i.e., the ones used to interact with the data sources); (ii) the processing units (in charge of security and data quality evaluation); (iii) the northbound interfaces (i.e., the publish/subscribe mechanism used for the sharing of the information with other services).

A. NOS southbound interfaces

NOSs collect data transmitted by different kinds of sources (i.e., nodes). The system has been designed to support both registered sources as well as anonymous ones, each characterized by different communication technologies and providing diverse quality levels for their data. As regards the registered nodes, NOSs provide a service for source registration by means of HTTP protocol; the related information are stored in the *Sources* data structure. Registered sources are associated with an identifier, and, optionally, with a geographical position and/or an encryption scheme, including the proper keys for interactions with NOSs. For each incoming data NOSs extract the following information: (i) the data source, which describes the type of node (the identifier in case of a registered source); (ii) the communication mode, which is the way in which the data is transmitted (e.g., discrete or streaming communication); (iii) the data schema, which represents the type (e.g., number, text) and the format of the transmitted data; (iv) metadata describing the data content; (v) a timestamp describing when the data was received by the NOS. The HTTP communication protocol is also used among NOS and the data sources for the data transmission. Since received data is highly heterogeneous, NOSs initially stores them in the *Raw Data* collection, and, periodically, elaborates them according to the two-phase process shown in Fig. 1 (including *Data Normalization* and *Analyzers*) in order to obtain an uniform representation and, as specified in the next section, enriching it with relevant metadata.

First, the data stored in *Raw Data* are put in the format specified in Fig. 2 by the *Data Normalization* module, which stores them in the *Normalized Data* collection. This represents a sort of pre-processing phase in which the unnecessary information is removed so to enable later processing stages to access the information in a unified way. Then, a second module, consisting of a set of *Analyzers*, periodically extracts the normalized data from the storage unit *Normalized Data* and elaborates them, computing relevant security and data quality indicators (see Sections III-B and III-C). Within such a step, as shown in Fig. 2, data is annotated with a set of metadata in the form of a score for each security and data quality property: data confidentiality, data integrity, source privacy and source authentication (security); data accuracy, data precision, information timeliness and completeness (data quality). The processed data is used for providing services to the target users (see Section III-D). Note that most of the existing approaches already address the issue of extracting data from heterogeneous sources, but very few focus on the analysis of security, privacy and data quality, as we do in this work.

As regards security and quality assessment, NOS provides two analyzers: *Security Analyzer* and *Quality Analyzer*, discussed in the following sections.

B. Data quality evaluation

For the data quality analysis, a score in the range $[0, 1]$ is assigned to timeliness, completeness, accuracy and precision levels [26],[27] by the *Quality Analyzer* (see Figure 1). *Timeliness* is defined as the temporal validity of data and is calculated on the basis of the freshness of data and on the frequency of data updates; it is usually measured as a function of two variables: currency and volatility:

$$Timeliness = \max\left(1 - \frac{Currency}{Volatility}, 0\right), \quad (1)$$

where $Currency$ is defined as the interval from the time when the value was sampled to the time instant at which data are received by the NOS. $Volatility$ is a static information which indicates the amount of time units (e.g., seconds) during which data remains valid; it is usually associated with the type of phenomena that the system has to monitor and depends on the timescale of its dynamics.

Completeness is calculated as the amount of collected values over a given time interval divided by the amount of expected values:

$$Completeness = \frac{collectedValues}{expectedValues}. \quad (2)$$

Note that missing values can be caused by sensors inefficiencies or communication issues.

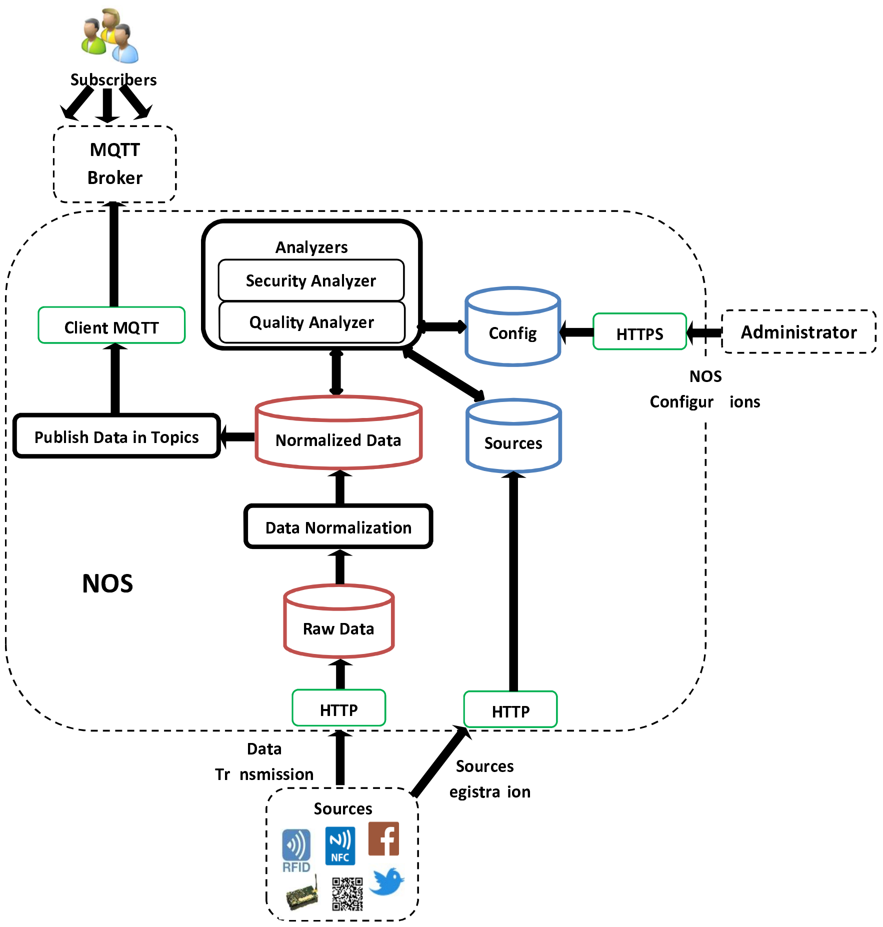

Fig. 1: System Architecture

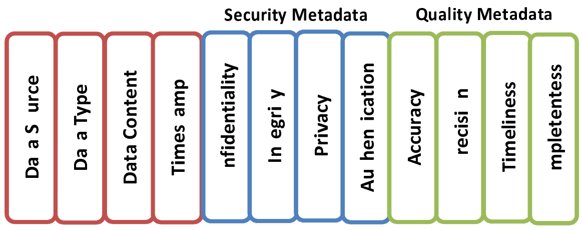

Fig. 2: NOS data format

Accuracy is usually defined as the degree of similarity of a measured quantity to its true value; it is also related to *precision*, which is the degree to which further measurement or calculations return the same or similar results. Ideally, a sensor shall be both accurate and precise, with all the measurements close to a reference value representing the true figure. In continuous value monitoring, accuracy and precision can be used as the features able to reveal errors or changes in the monitored process. Moreover, precision is often specified in terms of the standard deviation of the measured values: the smaller the standard deviation, the higher the precision. Formally, the accuracy of a value can be retrieved by calculating the error resulting from the difference between the sensed value v_n and a reference value v_{ref}. The acceptable measurement error can be defined as ϵ_{acc} and the measure is considered accurate if:

$$|v_n - v_{ref}| < \epsilon_{acc} \tag{3}$$

Considering such constraint, each value is associated with a boolean metadata: the value 1 is assigned to accurate values while the value 0 is assigned to inaccurate values. In this way, considering the streaming of values, the accuracy of the received values can be calculated as the ratio of the number of inaccurate values over the number of collected values:

$$Accuracy = 1 - \frac{wrongValues}{collectedValues}. \tag{4}$$

The precision can be defined as the inverse of variance. A measure is considered precise if:

$$1/n \cdot \sum_{n=1}^{N} (v_n - \mu)^2 < \epsilon_{prec} \qquad (5)$$

where μ is the average of the sequence v_n, while *N* is the maximum number of measurements considered in a specific time interval. Accordingly, an aggregate measure of precision can be computed as:

$$Precision = 1 - \frac{NotPreciseValues}{collectedValues}. \qquad (6)$$

Precision is a value that is mainly used to better understand the accuracy measure. Indeed, situations in which values are incorrect but precise should be thorough analyzed since inaccuracy can be caused by changes in the monitored phenomenon or by faulty sensors [26] .

Quality assessment is performed periodically on the basis of a window-based approach.

C. Security evaluation

For security purposes data sources are allowed to register to NOS; the registration gives various advantages, since it allows NOS to have a complete knowledge of the source itself and to establish an encryption scheme along with the proper keys and identifiers to be used. Some registered sources may use neither authentication credentials nor encryption. NOS accepts data also from anonymous sources; also in this case a security evaluation has to be performed. The *Security Analyzer* must be able to access to the *Sources* storage unit since, in order to analyze the received data, it may need information regarding the sources registered to NOS. To this purpose, NOS exploits an algorithm valid for both registered and anonymous sources, which aims to associate a score in the range $[0, 1]$ for the security metrics (i.e., confidentiality, integrity, privacy, authentication). As in the IoT context NOS may have to manage sensitive data, such security scores are intended as levels of *confidentiality* and *integrity* of the information transmitted to NOS, *privacy* of the transmitting source and *authentication* (i.e., the robustness of the source authentication towards NOS). Note that malicious devices may be represented by non-registered sources, which send violated data to NOS or execute malicious actions towards those transmitted by non-malicious ones (e.g., spoofing, sniffing).

In this paper, the security assessment algorithm takes into account two sets of parameters: a set of threats/attacks a_n and a set of security countermeasures c_m. The first one includes the attacks that may be carried out towards the sources or the data transmitted to NOS (e.g., data violation, unauthorized access, masking, impersonation). The second one regards the countermeasures made available by NOS in order to face the attacks included in a_n (i.e., encryption, authentication, key pre-distribution). The security model considered by the algorithm links the attacks of a_n with the corresponding countermeasures in c_m. The taxonomy of the security attacks and the related countermeasures is retrieved from [28]. This work is suitable for our needs to identify a set of attacks and countermeasures for the IoT environment, since it is not closely related to computers and networks threats, as most of the existing works on taxonomies, but focuses on embedded devices, which are strictly related to the IoT technologies. It refers to: (i) data confidentiality, authentication and integrity; (ii) data freshness and availability; (iii) key management protocols; (iv) reputation schemes. In detail it considers each countermeasure to present a degree of resistance to a violation or to an attack attempt. The relationship among attacks and countermeasures is many-to-many, because an attack can be tackled through a plurality of countermeasures and a countermeasure can face more than one attack. Each relationship is associated with a weight w_{a_i,c_i} in the range $[0 : 1]$, which represents the level of robustness of the countermeasure c_i with respect to the attack a_i (see Figure 3).

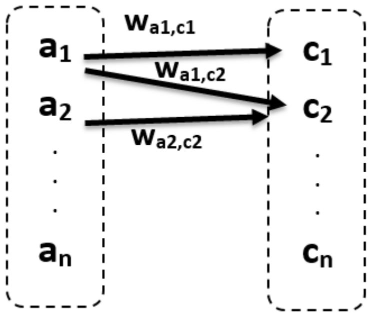

Fig. 3: Weighed relationships among attacks and countermeasures

The identified relationships are clustered into four groups, one for each security metric to be analysed (i.e., confidentiality, integrity, privacy, authentication). The assignment of each attack-countermeasure pair to a group is made by the NOSs administrators in an early phase of system configuration, but it can be updated at runtime. Hence, at this initial stage, there are four groups of sets of attacks-countermeasures, which are named as follows: (i) g_{conf} for attacks-countermeasures related to the data confidentiality; (ii) g_{int} for the pairs related to to data integrity; (iii) g_{pri} for privacy issues; (iv) g_{auth} for the pairs concerning sources authentication. Note that such groups are not necessarily disjoint, since a pair may belong to one or more groups, as shown in Figure 4. The information related to the groups are stored in the storage unit named *Config*, which will be described in detail in the following.

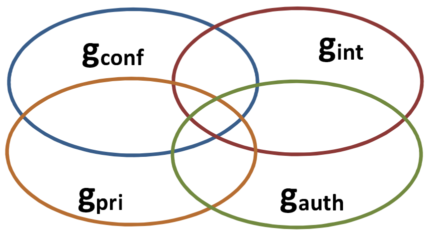

Fig. 4: Attacks-countermeasures groups

TABLE I: pairs attack-countermeasure

Attack	*Countermeasure*	*Group*
1) Packet sniffing	Data content encryption	g_{conf}
2) Password attack	Complex password generation	g_{conf}, g_{auth}
3) Man-in-the-middle attack	Data content encryption	g_{int}
4) Session hijacking attack	Secure session establishment	g_{int}, g_{auth}
5) Identity spoofing	Identity encryption	g_{pri}
6) Key impairment	Secure key distribution scheme	g_{conf}, g_{auth}

Table I shows some examples of attack-countermeasure pairs derived from the used taxonomy and classified in the groups described above. Note that in Table I the countermeasures are presented in a generic way, since, as regards, for example, data encryption, a source can adopt different encryption schemes (e.g., RSA, AES), and, as a consequence, the corresponding robustness level varies over the time. More in detail, there is a weight for each relationship among, for example, the man-in-the-middle attack and the different encryption schemes which can be adopted (e.g., man-in-the-middle attack and RSA, and attack man-in-the-middle and AES). The same considerations apply to password generation and secure session establishment techniques.

Once the groups are defined, NOSs can update the weight corresponding to the relationships among attacks and countermeasures depending on the sources behaviour and, consequently, on the basis of the received data. In this way, the system is able to evaluate the robustness of the countermeasures to particular kinds of attack during the normal IoT system operations, since NOS recognizes the possible malicious activities which can occur within the IoT system. When the NOS platform is deployed in an IoT context, the initial weights are conservatively all set to 1. During operations, a NOS is able to detect the following events: (i) data confidentiality and/or integrity violations; for example, a source may share some proper keys with NOS in order to encrypt its data; then, NOS may verify the integrity of a received data (if the data is recognized as violated, then an integrity attack to the encryption technique adopted by the source has been successful); (ii) sources privacy violations; (iii) unauthorized access to the system (e.g., password violation, so an unauthorized device has accessed the system); (iv) robustness of key management protocols in relation to key length (bits), deterministic or probabilistic generation of keys, encryption scheme adopted; (v) replay or routing attacks which can hinder the freshness and the availability of the data received from the different sources.

Weights can vary over time in a dynamic way; such variations depend on the events described above or on context changes such as a source changing the length of its keys or the encryption scheme adopted. Such a process of automatic adjustment is performed by means of a well-known learning approach, namely difference temporal learning [29]. It is suitable for the learning in dynamic environments and is able to make predictions about specific features on the basis of temporal differences observed during system activity. In this way, the weights may decrease over time with the observations of system or data violations, but they may also increase if a certain countermeasure turns out to be more resilient or if an attack is no longer performed. The following equation regulates the update of weights:

$$\Delta w_t = \alpha \cdot \sum_{k=1,t} \bigtriangledown_w \cdot w_t \tag{7}$$

where the weight variation Δw_t at time t depends on: (i) the learning rate constant α, which is a linear decreasing function of time; (ii) the sum $(\sum_{k=1,t} \bigtriangledown_w \cdot w_t)$, which is the sum of the gradients taking into account all the previous predictions until the time t. After the update, performed at time t on the i-th pair attack/countermeasure, the corresponding weight w_{t_i,c_i} is updated according to Eq. 8 (note that w corresponds to a weight w_{t_i,c_i}). Δw_t could assume negative values, but the resulting weights must be a value in the range $[0:1]$.

$$w_{t+1} = w_t + \Delta w_t \tag{8}$$

Once the attacks/countermeasures model is defined, the algorithm computes for each incoming data the related security scores on the basis of the actual weights and of the data source src. Equations 9, 10, 11 and 12 show how the score corresponding to the level of confidentiality (sec_{conf}), integrity (sec_{int}), privacy (sec_{pri}) and authentication (sec_{auth}), respectively, is determined.

$$sec_{conf} = \frac{a_{conf,src}}{a_{g_{conf}}} \cdot \frac{\sum\limits_{i \epsilon a_{conf,src}, j \epsilon c_{conf,src}} w_{i,j}}{c_{conf,src}} \tag{9}$$

where: $a_{conf,src}$ is the number of confidentiality attacks which the source src could suffer; $a_{g_{conf}}$ is the total number of attacks included in the model in the group g_{conf} for any kind of sources (not only those related to src); $c_{conf,src}$ is the number of countermeasures adopted by src related to the attacks to confidentiality included in $a_{conf,src}$. The sum of the weights considers only the weights between the attacks in $a_{conf,src}$ and the countermeasures in $c_{conf,src}$. For example, suppose that the source src adopts AES for encrypting its data; moreover, it also adopts a 8-bit length password as credential for ensuring both confidentiality and authentication. As shown in Table I (points 1 and 2), AES is a countermeasure associated to the g_{conf} group; while the password is associated to both g_{conf} and g_{auth} groups. The steps performed by NOS to assess over the time the confidentiality score sec_{conf} are the following:

- The initial weights corresponding to the two pairs attack-countermeasure (i.e., AES-packet sniffing, 8-bit password-credential violation) are set to 1; therefore, the corresponding confidentiality score sec_{conf} is 1. Eq. 9 is initially evaluated as shown in Figure 5. For simplicity, $a_{conf,src}$ is considered, in this example, equal to $a_{g_{conf}}$. $a_{conf,src}$ is composed by two elements (i.e., packet sniffing and credential violation), and also $a_{conf,src}$ (i.e., AES and 8-bit password).

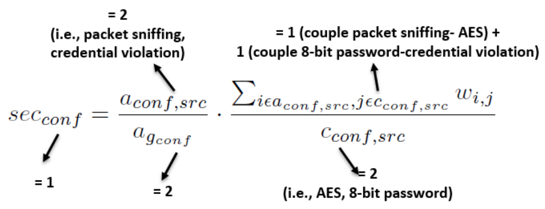
= 2
(i.e., packet sniffing, credential violation)
= 1 (couple packet sniffing- AES) + 1 (couple 8-bit password-credential violation)
$sec_{conf} = \frac{a_{conf,src}}{a_{g_{conf}}} \cdot \frac{\sum_{i \epsilon a_{conf,src}, j \epsilon c_{conf,src}} w_{i,j}}{c_{conf,src}}$
= 1
= 2
= 2
(i.e., AES, 8-bit password)

Fig. 5: Confidentiality score assessment - initial stage

- During the system operations, NOS recognizes no violated packets from the source src, but several times its password has been intercepted (e.g, through brute-force attack).
- As a consequence, the weight related to the pair 8-bit password-credential violation decreases; for such an example, it is updated to 0.3 by the learning algorithm (Eq.s 7 and 8).
- The new data obtained from the source src will receive a lower confidentiality score sec_{conf}, which is recomputed to 0.65, as shown in Figure 6.

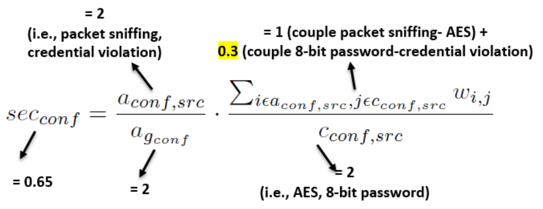
= 2
(i.e., packet sniffing, credential violation)
= 1 (couple packet sniffing- AES) + 0.3 (couple 8-bit password-credential violation)
$sec_{conf} = \frac{a_{conf,src}}{a_{g_{conf}}} \cdot \frac{\sum_{i \epsilon a_{conf,src}, j \epsilon c_{conf,src}} w_{i,j}}{c_{conf,src}}$
= 0.65
= 2
= 2
(i.e., AES, 8-bit password)

Fig. 6: Confidentiality score assessment - update

As a consequence, a user who wants to receive data from the source *src* will be aware that they have a level of confidentiality not greater than 0.65, so there is a 35% risk of a confidentiality attack.

Instead, considering a malicious device which tries to execute a man-in-the-middle attack among two registered sources src_1 and src_2, the scope of the proposed security algorithm is to evaluate how robust is the countermeasures adopted by such sources (i.e., the robustness of the data content encryption scheme adopted, as shown in Table I). For example, in this case, the affected score is that of integrity sec_{int}. Therefore, NOS executes the following steps:

- The initial weights corresponding to the attack-countermeasure pairfor the two sources, src_1 and src_2, are both set to 1, but src_1 adopts a 128-bit key for encrypting its data, while src_2 uses a 256-bit key.
- If NOS analyzes the level of robustness of such a countermeasure in terms of integrity against a possible man-in-the-middle attack, then it considers the difference in the bit-length of the used keys, and, following Eq. 10, the result will be that $sec_{int_{src_1}}$ is less than $sec_{int_{src_2}}$.

Therefore, a user may choose to receive data only from the source src_2, since it presents a greater level of integrity with respect to the source src_1.

$$sec_{int} = \frac{a_{int,src}}{a_{g_{int}}} \cdot \frac{\sum_{i \epsilon a_{int,src}, j \epsilon c_{int,src}} w_{i,j}}{c_{int,src}} \tag{10}$$

The same considerations apply to the following metrics:

$$sec_{pri} = \frac{a_{pri,src}}{a_{g_{pri}}} \cdot \frac{\sum_{i \epsilon a_{pri,src}, j \epsilon c_{pri,src}} w_{i,j}}{c_{pri,src}} \tag{11}$$

$$sec_{auth} = \frac{a_{auth,src}}{a_{g_{auth}}} \cdot \frac{\sum_{i \epsilon a_{auth,src}, j \epsilon c_{auth,src}} w_{i,j}}{c_{auth,src}} \tag{12}$$

Note that, as stated above, the presented algorithm is suitable for both registered and non-registered sources. It is very remarkable that the proposed algorithm allows to perform a security analysis and obtain a valid score (not trivially set to 0 or to "undefined") also for non-registered sources, with which NOS does not share, for example, any encryption schemes. This is achieved by analysing the data they provide over time and the node's behaviour within the IoT system. Concluding the discussion about security evaluation, Algorithm 1 summarizes the steps performed by the security assessment scheme proposed in this paper. Note that NOS is conceived as a modular architecture, therefore such a mechanism can be enabled or disabled on the basis of

Algorithm 1 Security Assessment

```
Input: $Relationships\ among\ a_n,\ c_n\ from\ the\ taxonomy$
1: for all $a_i, c_i$ do
2:     $Assigment\ to\ groups\ g_{conf},\ g_{int},\ g_{pri},\ g_{auth}$
3:     $Initialization\ of\ the\ weight\ w_{a_i,c_i}\ to\ 1$
4: end for
5: while $Learning\ is\ enabled$ do
6:     $Events\ monitoring$
7:     $Weights\ adjustment\ (Eqs\ 7\ and\ 8)$
8:     $Scores\ update\ (Eqs\ 9,\ 10,\ 11,\ 12)$
9: end while
```

the current application needs. The effectiveness of the proposed algorithm will be discussed in Section V.

The rules just presented for the assessment of the security and quality scores of data are stored in a proper format in another NOS storage unit called *Config*. Such a collection contains all the configuration parameters required for the correct management of the IoT system (e.g., how to calculate quality properties, which attacks or security countermeasures to consider), represented in JSON format (as described in Section IV). It can also be configured at any time by an IoT system administrator through a secure connection (e.g., via HTTPS) depending on the requirements of the specific deployment, without the need to re-start the NOS system. The communication protocol to be used is HTTPS, since the policy adopted by the NOS for processing the IoT data have to be protected against external attacks. *Analyzers* periodically query the *Config* storage unit in order to know which rules shall be used.

The NOS architecture has been designed to provide a score for each security and data quality requirement; in this way a possible application scenario can be easily integrated depending on its purposes and on the specific context, and can benefit of a high level of flexibility. For example, some applications require to use data with a high level of integrity and confidentiality, but there is no interest in privacy issues. Other application domains, on the contrary, may aim to provide a service characterized by error-free data and high confidentiality scores, therefore the data to be selected are those provided by sources able to satisfy these requirements. Note that searching and selecting data sources when there is no description neither about the sources nor the acquired data is a very challenging task. Although a NOS is not able to directly counteract malicious devices, it is able to recognize that the data provided by a source is corrupted or it presents a poor level of confidentiality or privacy and discard it as unsuitable. Note that, after repeatedly receiving bad data from the same source, NOS could even block it. By means of the algorithm presented above, a NOS is able to perform automatic reasoning about data quality and security and, then, allow the users to filter information and deal properly with the massive amount of data received by IoT services.

D. NOS northbound interfaces

While the NOSs southbound interfaces are based on the HTTP protocol, the northbound interface is based on the Message Queue Telemetry Transport (MQTT) protocol. MQTT is a lightweight publish/subscribe protocol [30] specifically designed for resource-constrained devices. In the IoT context, it is widely used to enable communication among devices using a publish/subscribe messaging approach. An MQTT client, as that contained in NOSs, exchanges messages with an MQTT broker by means of publications and subscriptions to topics. Such mechanism is adopted to support interactions among services and IoT devices. NOSs include a module in charge of assigning data items to the corresponding topic and to publish them on a MQTT broker, as depicted in Fig. 1. The mapping of data to a specific topic depends on the application domain and is out of the scope of this work, but it may require the usage of an ontology able to represent the semantic of the managed resources. In general, topics are multi-level structures separated by a forward slash similar to a directory structure. An example of a topic for publishing temperature information of a sensor with identifier *sensorId* could be *sensor/temperature/sensorId*. Note that subscribers may register for specific topics at runtime and NOSs provide a mechanism for dynamic subscription and unsubscription to topics. According to the MQTT protocol, messages can be published with a Quality of Service (QoS) parameter indicating that a message should be delivered "at most once", "at least once" and "exactly once". MQTT also supports persistence of messages to be delivered to future clients that subscribe to a topic, and may be configured to send messages of specific topics when the subscriber connection is abruptly closed. These parameters are is specified in the *Config* storage unit. Summarizing, a typical MQTT message includes the following parameters: (i) the topic; (ii) the data value; (iii) the QoS level; (iv) the retain value.

IV. Prototype

The NOS system presented in Section III has been implemented as a prototypical service middleware platform able to manage a large amount of data from heterogeneous devices with lightweight modules and interfaces working in a non-blocking manner to perform data analysis, discovery, and query [5]. This represents an important step beyond conventional ad hoc centralized IoT solutions. In a real scenario, one or more NOSs can be deployed in a distributed manner. Note that, from an analysis of NOS functionalities, there is no need for a peer-to-peer management of NOSs. In fact, NOSs are able to: (i) independently handle the connected data sources, without the need to inform other NOSs of their active and past interactions; (ii) be independently re-configured by IoT system administrators through the *Config* interface; (iii) independently assign topics and publish data on the basis of the defined rules. Note that the existing IoT deployments are often hardly reconfigurable [31], since they are conceived for very specific applications, based on a vertical silo-based approach. The middleware proposed in this work supports dynamic reconfiguration and can be remote orchestrated through Internet/intranet protocols, which are based on open standards (see Section III-D).

The NOS prototypical implementation is compliant with the architecture presented in Section III. The *Node.JS* platform[1] has been used for the platform implementation, *MongoDB*[2] for storage management, and Mosquitto[3] for the publish/subscribe system. The code is released as open source under a permissive license[4]. Modules interact among themselves via *RESTful* services. They can be distinguished in: node interfaces, processing modules, and service interfaces.

The node interfaces manage the sources registrations and receive the data from IoT devices. As described in Section III-A, these correspond to the southbound interfaces of NOS. The following endpoints are exposed:

- **POST data/** Used for handling transmission of data from the nodes to the NOS. Messages shall be formatted as valid JSON nodes.
- **POST registration/** Used by nodes for registering to the NOS. Messages shall be formatted as valid JSON nodes. The following fields are mandatory: `NodeId`, `NodeType`, `CommunicationMode`. Optional field: `EncryptionScheme`. The response includes node credentials.

The NOS system includes also additional modules, namely data normalization and data analyzers. Such processing modules are daemons which start along with the NOS and periodically extract data from *Raw Data* or *Processed Data* storage units. Since they are internal to NOS, no specific APIs are provided.

As far as northbound interfaces are concerned, the following endpoint is exposed:

- **mqtt.Client#publish(topic,payload,[options])** Used by NOS for publishing normalized data to the MQTT broker. Mandatory parameters: `topic` (channel on which the message is to be published), `payload` (message to be published). Optional parameters: `[options]` (QoS, retain flag, callback). The address of the broker is specified at client initialization time.

A great advantage of our approach is that it is possible to introduce new modules or duplicate the existing ones since they are able to work in a parallel and non-blocking manner. Moreover, it is possible to add new functionality or to remove active ones without re-starting the whole system. The use of the non-relational *MongoDB* database allows the data model to evolve dynamically. Furthermore, such an implementation is independent both of the data model and the application domain.

Given the overall system architecture, it is worth remarking that NOS does not require data persistence for IoT-generated data. Rather, data is temporarily cached on the NOS while being processed and before being submitted to the MQTT broker. Accordingly, NOS uses the in-memory capability of *MongoDB* for two of its databases, namely *Raw Data* and *Normalized Data*, whereas the databases *Config* and *Sources* must be persistent. A routine runs on NOS in order to remove from *Raw Data* the data already normalized and from *Normalized Data* the data already published. Such an approach greatly improves NOS performance, significantly lowering query and read/write performance.

As far as the MQTT broker is concerned, the open-source *Mosquitto* implementation —compliant with version 3.1.1. of the MQTT protocol— is used. Note that the broker is a module which runs out of the NOS system, therefore it acts as an intermediary among NOS and the subscribers.

V. Experimental Validation

In our experimental setup, the NOS platform runs on a Raspberry Pi, which has the computational and resources capabilities for running the NOS functionalities. To simulate the behaviour in a real-world setting, the NOS is connected to a number of open data feeds. In particular we use data provided in real time from six sensors at the meteorological station of the city of Campodenno (Trentino, Italy). Data refers to temperature, humidity, wind speed, energy consumption and air quality; a web service exposes them in *JSON* format, and the NOS retrieves them through HTTP *GET* requests, as detailed in Section IV. According to the prototypical implementation presented in the previous sections, the NOS processes data assessing their reliability in terms of security, privacy and quality. Data is then transmitted to a MQTT broker. A simple visualization service is also implemented, which makes it possible for users to set their preferences in terms of security, privacy and data quality and to visualize compound indicators computed only from data matching the expressed preferences.The dashboard is shown in Figure 7.

[1]`http://nodejs.org/`
[2]`http://www.mongodb.org/`
[3]`http://mosquitto.org`
[4]`https://bitbucket.org/alessandrarizzardi/nos`

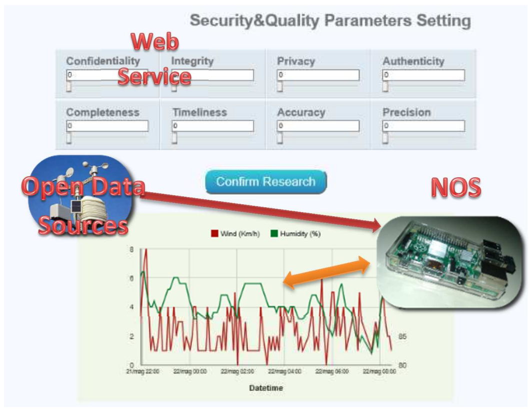

Fig. 7: User Dashboard

TABLE II: Sources parameters

Sources	*Authentication*	*Encryption technique*
Source 1	IPSec	RSA
Source 2	16-bit password logon	AES
Source 3	none	MD5
Source 4	8-bit password logon	SHA-1
Source 5	none	none
Source 6	8-bit password logon	none

Since meteorological data are obtained from six different sources in real time, it is assumed that each of them adopts different authentication and encryption methods for communicating with the NOS, as summarized in Table II.

The evaluation of the the data provided, both in terms of security and quality, is performed by the system following the methods presented in Sections III-B and III-C, eventually returning a score for each security and quality metric. For testing the effectiveness of the proposed mechanisms, the system has been observed for a period of a week. Figures 8 (for security) and 9 (for data quality) show the day-by-day evaluation regarding the data provided by the six sources. For security assessment, the outcomes are compliant with the expected robustness of the authentication and encryption techniques adopted by the monitored sources. Note that each score is initially set to the maximum value (i.e., 1), as specified in Sections III-B and III-C, and that a source may change its communication agreement with NOS, thus modifying the corresponding security assessment scores. From the figures, it can be observed that NOS can interact which sources characterized, for example, by a good level of authentication, but, at the same time, by a low level of reliability in terms of confidentiality, integrity and privacy (e.g., source 6 in Figure 8) and vice versa (e.g., source 3 in Figure 8). Another case is that the data provided by a source may present good levels of completeness and timeliness, but poor accuracy and precision levels (e.g., source 4 in Figure 9). The assessment confirms the potential of the NOS approach in empowering the user with the ability to choose the requirements to be met by the data used by a given IoT-enabled service.

More in detail, in order to clarify the effectiveness of the new security evaluation algorithm proposed in Section III-C, Figure 10 shows an analysis of the behaviour of source number 6 with respect to the integrity score in particular conditions. From Figure 8 it emerges that such source obtains low scores for data integrity (at day 7 the integrity score is equal to 2). Now, the whole set of attack-countermeasure pairs, referred to the integrity of the data of source number 6, are considered and remain constant for the whole observation period, except for one pair, which corresponds to a man-in-the-middle-attack able to modify the data content and, therefore, to violate the integrity of the information transmitted by the source node. As reported in Table II, source number 6 does not adopt any encryption technique, therefore the man-in-the middle attack simulated within the network is successful. As a consequence, the integrity score gets lower. However, if the source decides at day 2 to adopt AES for encrypting its data, sharing the proper keys with NOS, it is expected that the security algorithm varies the integrity evaluation accordingly. In fact, Figure 8 represents the integrity score assessment in the "normal" case (i.e., the source does not change its communication agreements with NOS) and in the "modified" case. Note that, at day 6, source number 6 revokes the use of the encryption scheme with NOS sending again the data in clear; as shown in Figure 8, the integrity score starts again to get lower.

The presented results just represent an example of how NOS interacts with IoT data sources. Moreover, it shows how NOS, although it does not directly tackle attacks able to compromise the IoT devices, it recognizes the possible threats for each data source and, consequently, it estimates the corresponding levels of security. In this way, users can select the data with a deep level of awareness about the services offered by the IoT system. Our plan for the next future includes a consistent evaluation in different domains and IoT contexts, in particular including an analysis of the behaviour in the presence of sensitive data and of nodes joining and leaving the system.

VI. Conclusions

The rising adoption, at different levels and within different application domains, of IoT technologies is fostering requests for solutions able to harness the heterogeneity of IoT devices while at the same time, capable of guaranteeing an adequate level of security and data

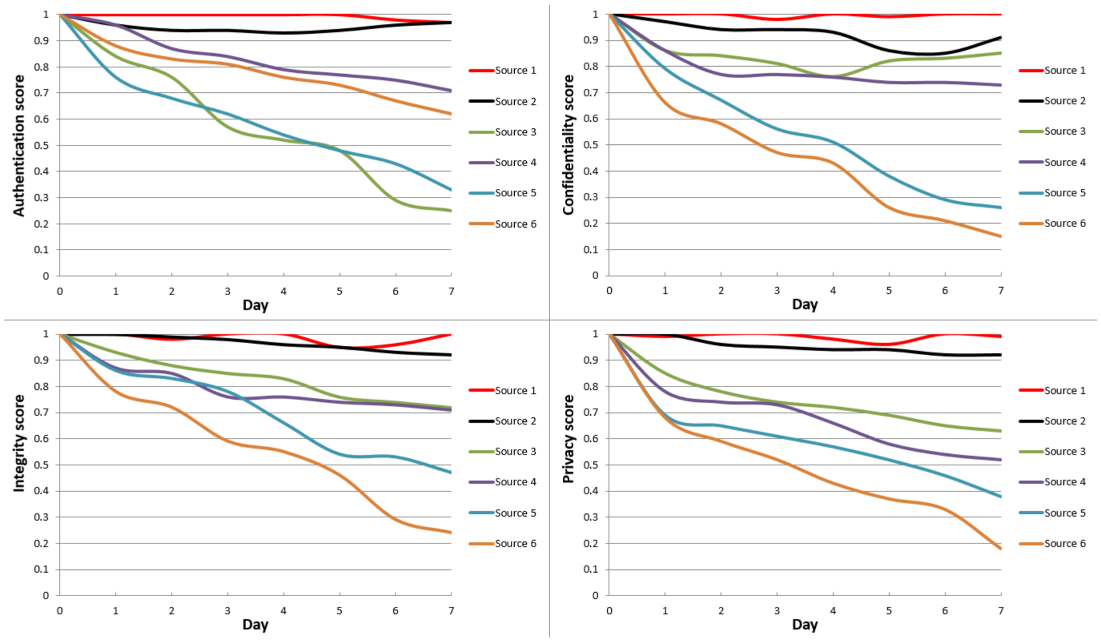

Fig. 8: Security Score Evaluation

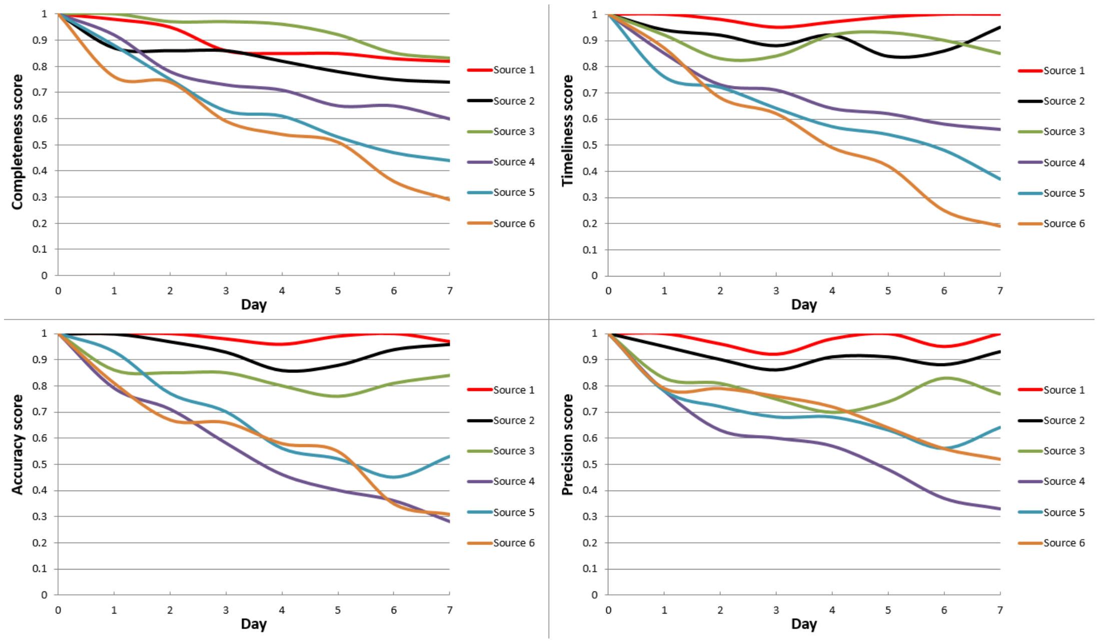

Fig. 9: Quality Score Evaluation

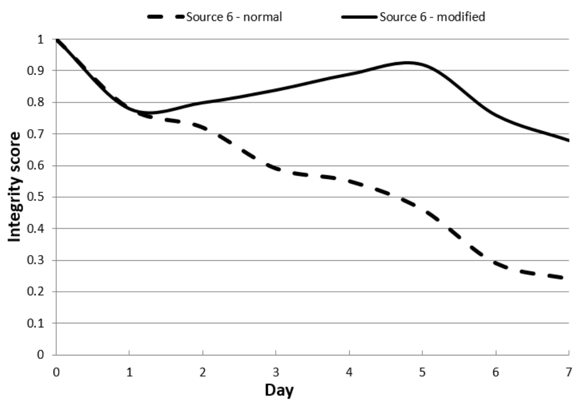

Fig. 10: Attack-Countermeasure Analysis

quality used to power services and applications. In this paper we have presented the design and a prototypical implementation of a distributed middleware layer, named NOS, able to manage heterogeneous data sources, to provide a uniform, consistent data representation and to evaluate the security and quality level associated to each data unit. In particular, a proper security algorithm has been developed in order to assess the trustworthiness of registered and non registered IoT data sources. The effectiveness of the proposed solution has been validated through the implementation of a real prototype of the NOS platform. Future extensions include the introduction of a key management system in the platform.

REFERENCES

[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, 2015.

[3] A. Coen Porisini, P. Colombo, and S. Sicari, *Privacy aware systems: from models to patterns*, igi global ed. Software Engineering for Secure Systems: Industrial and Research Perspectives, 2011.

[4] S. Sicari, A. Rizzardi, C. Cappiello, and A. Coen-Porisini, "A NFP model for internet of things applications," in *Proc. of IEEE WiMob*, Larnaca, Cyprus, Oct 2014, pp. 164–171.

[5] A. Rizzardi, D. Miorandi, S. Sicari, C. Cappiello, and A. Coen-Porisini, "Networked smart objects: Moving data processing closer to the source," in *2nd EAI International Conference on IoT as a Service*, Oct 2015.

[6] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, Nov 2007.

[7] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and managing web services: Issues, solutions, and directions," *The VLDB Journal*, vol. 17, no. 3, pp. 537–572, May 2008.

[8] "Peertrack," http://cs.adelaide.edu.au/peertrack/.

[9] "PERCI (PERvasiveServiCe interaction)," http://www.hcilab.org/projects/perci/index.htm.

[10] M. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the Internet of (important) Things," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 3, pp. 1389–1406, Third 2013.

[11] I. Bagci, S. Raza, T. Chung, U. Roedig, and T. Voigt, "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, June 2013, pp. 523–631.

[12] D. Boswarthick, O. Elloumi, and O. Hersent, *M2M Communications: A Systems Approach*, 1st ed. Wiley Publishing, 2012.

[13] D. Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. Spirito, "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, July 2012, pp. 1–6.

[14] A. Gòmez-Goiri, P. Orduna, J. Diego, and D. L. de Ipina, "Otsopack: Lightweight semantic framework for interoperable ambient intelligence applications," *Computers in Human Behavior*, vol. 30, pp. 460–467, January 2014.

[15] C. H. Liu, B. Yang, and T. Liu, "Efficient naming, addressing and profile services in Internet-of-Things sensory environments," *Ad Hoc Networks*, vol. 18, no. 0, pp. 85–101, 2013.

[16] S. Sicari, C. Cappiello, F. D. Pellegrini, D. Miorandi, and A. Coen-Porisini, "A security-and quality-aware system architecture for Internet of Things," *Information Systems Frontiers*, pp. 1–13, 2014.

[17] "European FP7 IoT@Work project," http://iot-at-work.eu.

[18] "iCORE project," http://www.iot-icore.eu.

[19] "IOT-EST project," http://ict-iotest.eu/iotest/.

[20] "EBBITS project," http://www.ebbits-project.eu/.

[21] "Usable trust in the Internet of Things," http://www.utrustit.eu/.

[22] "BUTLER project," http://www.iot-butler.eu.

[23] B. Guo, D. Zhang, Z. Wang, Z. Yu, and X. Zhou, "Opportunistic iot: Exploring the harmonious interaction between human and the internet of things," *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1531–1539, Nov. 2013.

[24] A. Metzger, C.-H. Chi, Y. Engel, and A. Marconi, "Research challenges on online service quality prediction for proactive adaptation," in *Software Services and Systems Research - Results and Challenges (S-Cube), 2012 Workshop on European*, June 2012, pp. 51–57.

[25] F. Li, S. Nastic, and S. Dustdar, "Data quality observation in pervasive environments," in *Proceedings of the 2012 IEEE 15th International Conference on Computational Science and Engineering*. IEEE Computer Society, 2012, pp. 602–609.

[26] C. Cappiello and F. A. Schreiber, "Quality- and energy-aware data compression by aggregation in WSN data streams," in *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–6.

[27] A. Klein and W. Lehner, "Representing data quality in sensor data streaming environments," *J. Data and Information Quality*, vol. 1, no. 2, pp. 10:1–10:28, Sep. 2009.

[28] T. Roosta, S. Shieh, and S. Sastry, "Taxonomy of security attacks in sensor networks and countermeasures," in *The first IEEE international conference on system integration and reliability improvements*, vol. 25, 2006, p. 94.

[29] G. Tesauro, *Practical issues in temporal difference learning*. Springer, 1992.

[30] "IBM and eurotech, "mqtt v3.1 protocol specification"," http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html.

[31] I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, and D. P. Agrawal, "Choices for interaction with things on Internet and underlying issues," *Ad Hoc Networks*, vol. 28, no. 0, pp. 68–90, 2015.