IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 7, 2020, accepted January 19, 2020, date of publication January 22, 2020, date of current version February 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2968741

Smart City IoT Platform Respecting GDPR Privacy and Security Aspects

CLAUDIO BADII, PIERFRANCESCO BELLINI, ANGELO DIFINO, AND PAOLO NESI, (Member, IEEE)
Department of Information Engineering, University of Florence, 50121 Florence, Italy
Corresponding author: Paolo Nesi (paolo.nesi@unifi.it)

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Program under Agreement 688196.

ABSTRACT The Internet of Things (IoT) paradigm enables computation and communication among tools that everyone uses daily. The vastness and heterogeneity of devices and their composition offer innovative services and scenarios that require a new challenging vision in interoperability, security and data management. Many IoT frameworks and platforms claimed to have solved these issues, aggregating different sources of information, combining their data flows in new innovative services, providing security robustness with respect to vulnerability and respecting the GDPR (General Data Protection Regulation) of the European Commission. Due to the potentially very sensible nature of some of these data, privacy and security aspects have to be taken into account by design and by default. In addition, an end-to-end secure solution has to guarantee a secure environment at the final users for their personal data, in transit and storage, which have to remain under their full control. In this paper, the Snap4City architecture and its security solutions that also respect the GDPR are presented. The Snap4City solution addresses the full stack security, ranging from IoT Devices, IoT Edge on premises, IoT Applications on the cloud and on premises, Data Analytics, and Dashboarding, presenting a number of integrated security solutions that go beyond the state of the art, as shown in the platform comparison. The stress test also included the adoption of penetrations tests verifying the robustness of the solution with respect to a large number of potential vulnerability aspects. The stress security assessments have been performed in a piloting period with more than 1200 registered users, thousands of processes per day, and more than 1.8 million of complex data ingested per day, in large cities such as Antwerp, Helsinki and the entire Tuscany region. Snap4City is a solution produced in response to a research challenge launched by the Select4Cities H2020 research and development project of the European Commission. Select4Cities identified a large number of requirements for modern Smart Cities that support IoT/IoE (Internet of Things/Everything) in the hands of public administrations and Living Labs, and selected a number of solutions. Consequently, at the end of the process after 3 years of work, Snap4City has been identified as the winning solution.

INDEX TERMS End-2-end, GDPR, IoT, security, smart city.

I. INTRODUCTION

IoT (Internet of Thing) is becoming a disruptive technology, especially for city users of metropolitan areas. The pervasiveness of IoT Devices, integrated in common objects, is becoming increasingly deeper. The addresses' space for these devices would be enough to point any sensors of any devices at any moment without restrictions. Diffuse products that implement *Low-Power Wide Area Networks* (LPWAN) technologies for IoT introduced by SigFox and Semtech (LoRa, Long Range) have been gaining interest and have been under intense deployment campaigns worldwide [1]. At the same time, *short range* IoT devices (based on technologies such as IEEE 802.15.4 or Bluetooth Low Energy, BLE, [2]) are sold in increasing quantities and are already able to support scenarios for smart homes, energy metering and industrial automation. On the other hand, the start of the diffusion of *5G devices* and services is creating high expectations in networking IoT technologies, as the killer application of previous technologies in metropolitan areas.

The associate editor coordinating the review of this manuscript and approving it for publication was Adnan M. Abu-Mahfouz.

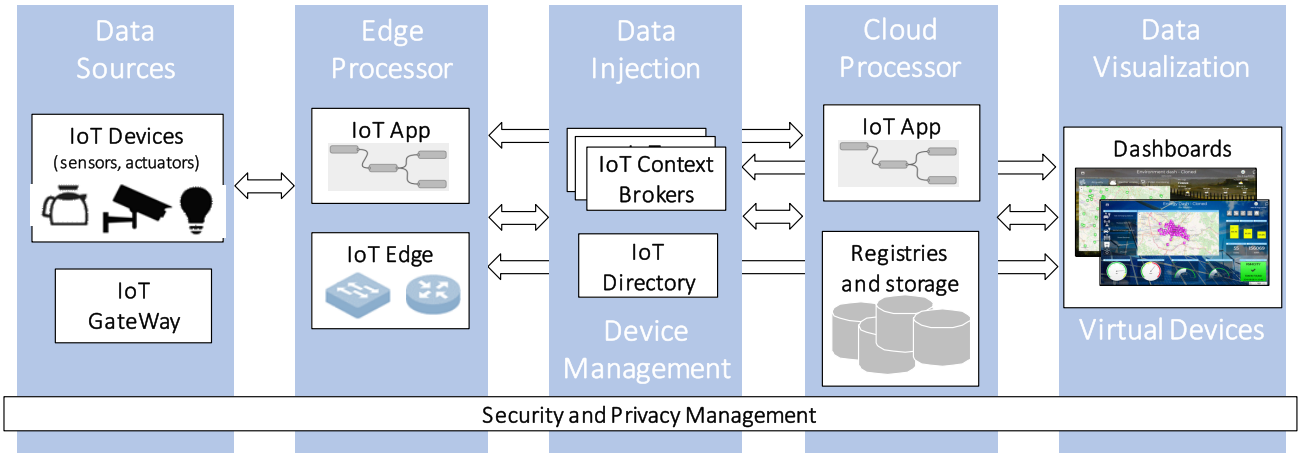

FIGURE 1. IoT platform general architecture.

With a high peak data rate, high connection density and better network energy efficiency, the 5G standard also addresses ad hoc specification for the M2M (Machine 2 Machine) use cases. With the help of the IPv6 (Internet Protocol version 6) address schema, the 5G standard will be capable to deliver a complete integration of IoT Devices [3].

A wide variety of objects (e.g., smart bulbs, IP (Internet Protocol) cameras and alarm clocks) are currently part of the user's environment, relying on the modem and hub devices of the user homes to connect them over the Internet. The IoT infrastructures permit the realization of a more integrated scenario where real world and smart devices complete each other and permit management with less human intervention: open communication schema supporting different standard protocols (e.g., MQTT over TLS, Message Queuing Telemetry Transport over Transport Layer Security) enable devices to connect each other and to exploit cloud-fog infrastructures [4]. In terms of security and management of data privacy, the complexity is growing, not only for the needs of establishing secure connection but also for the data management and access rules [5].

In most cases, the communications from IoT Devices and *IoT Brokers* is established only by using simple and not mutual authentications. The IoT Brokers are gateways in which the IoT Devices are registered, and by which they can be managed (for example, for firmware updates, for sending massive reset commands, etc.). In addition, IoT Devices and IoT Brokers could communicate with *IoT Edge Devices* that are entitled to perform some computation up to machine learning [6]. IoT Edge Devices are typically located on premise and have additional complexity in terms of secure communication with other tools on the cloud. In fact, IoT Edge solutions typically need to send/get data on/from the cloud, and directly on/from dashboards [7]. Moreover, IoT Edge devices could also be enabled to manage/process private data. Most of the IoT frameworks on the cloud provide structures and processes that may coordinate and control several different IoT elements, in the field and on the cloud as well. In certain cases, the IoT solution may be limited to on-premise solutions and tools without the need of having a cloud counterpart.

The IoT frameworks may present a set of modules, rules and algorithms that organize the ways in which data processing is performed and managed among involved entities (i.e., devices, edges, processes on cloud/containers, dashboards and, users). Even more, *IoT Platforms* may go beyond frameworks and may need the implementation of the so-called *IoT Applications* that permit definition of user-component-logic, at the same time, hiding the complexity of the infrastructure (of the IoT Frameworks) [8]. Furthermore, the IoT Application processes may manage open and private data and may produce results that are of a private nature of some user(s). Please note that IoT Applications may also be deployed on IoT Edge Devices (see Figure 1).

The information sent/received to/from the IoT Devices could be very sensible and private; therefore, protection and cryptography techniques are diffusely implemented. All steps of a secure communication must be guaranteed as well as solutions to protect data during all of the life cycle, including communicating, consumption for data analytics, up to the visualization [10], and acting back on actuators. It has to be remarked that machine to machine communications among IoT elements have to be secure, without the need of having them personally authenticated/authorized by the owner since in most cases the user does not know that data are processed by some process in the cloud or on the IoT Edge. In most of the scenarios, a high level of *security* has to be assured since the users rely on the system to manage private data, which can be eventually exploited to help-inform-assist the data owner in daily activities and tasks [12].

The IoT Platform architectures have to offer a high level of security. They need to support a *heterogeneous source* of information (IoT Devices, sensors/actuators, mobile, and streams) that is accessed in a multitude of different manners,

and by using *various communication systems* in scenarios where in some cases, it is difficult to predict when and where the data are generated and made available for the system/subsystems of the platform [13]. For these reasons, the architecture of secure IoT platforms must be carefully designed, due to the huge amount of data exchanged among parties, the complexity and the heterogeneity of the protocols and devices involved [9], and the level of security assured to the users [11].

Moreover, a layer of complexity has been added by the final adoption of the European Union General Data Protection Regulation 2016/679 (GDPR) [66]. The regulation was proposed in April 2016 and became operative in May 2018. It has been designed to improve the former European Union Data Protection Directive 95/46/EC (introduced in October 1995), with the main goals to (a) provide *uniform* guidelines about the protection of individuals with regard to the processing of personal data and on the free movement of such data *in all the EU (European Union) state members* and (b) provide *adequate* recommendations with respect to the *technology enhancement* that occurred over the past 20 years. According to the GDPR, specific mechanisms to save and manage authentications and data messages have to be taken into account by design and by default. It resulted in a strong impact on the implementation of complete IoT end-to-end stacks, as explained in this paper. Please note that GDPR aspects are more relevant in IoT applicative domains in which several users are involved, as in the Smart City, Smart House, Smart Health, etc., whereas they are less relevant in Industry 4.0 solutions in which all of the data have to be kept private and are typically owned by a single owner/industry and are rarely in need of differentiated accesses among users/groups at a fine-grained level [63]. This may happen in the cases in which the control based on Industry 4.0 IoT is performed on some cloud implementing an ''*Industry 4.0 as a Service*'' solution. Moreover, the compliance of an IoT-based solution with GDPR is a very complex aspect to be assessed [64]. There are a large number of online tools for assessing the compliance to GDPR that are applicable to institutions since the GDPR includes guidelines for managing privacy and not only guidelines and features for solutions. There is not a unique recipe or checklist to assess that a web application is GDPR-compliant and neither that an IoT Application is compliant, which is a much more complex architecture.

In this paper, the privacy and security issues related to the design of the Snap4City IoT Framework are addressed (https://www.snap4city.org). Snap4City is a solution produced in response to a research challenge launched by the **Select4Cities** PCP (Pre-Commercial Procurement) H2020 research and development project of the European Commission (https://www.select4cities.eu). Select4Cities identified a large number of functional (mainly Smart City IoT) and nonfunctional requirements (e.g., open source, scalability, security, GDPR compliance, modularity, usability, and working on the cloud and on premise) which are reported on their web site, and aimed at creating the best solution for modern Smart Cities supporting IoT/IoE in the hands of public administrations and supporting creation of collaborative Living Labs (thus stressing the aspects of GDPR and security). Most of the identified requirements have been taken from the large association of Living Lab ENOLL (European Network of Living Lab association, https://enoll.org), and from concertation of smart cities at level of European Commission as EIP-SCC (European Innovation Partnership on Smart Cities and Communities, https://eu-smartcities.eu).

Snap4City responded to the research challenge and with more than 14 months of work, developed and set an operative solution, which has been demonstrated to satisfy all Select4Cities requirements, including those regarding GDPR and security discussed in this paper. Snap4City, in brief, enables the creation and management of communities of users that collaboratively realize the following: (i) create IoT solutions and are somehow connected to organizations (cities, regions, industries, group of users, and final users as well), (ii) exploit open and private data with IoT and IoE devices respecting GDPR, and (iii) create/use processes and IoT Applications that could be on the IoT edge, mobile and cloud and they may interact with each other and with users via messages, dashboards and applications of different kinds. The produced Snap4City solution is GDPR-compliant and provides end-2-end secure connections on the IoT stack. The set of highly challenging requirements constrained the Snap4City team to address the abovementioned technical and scientific unaddressed problems, most of them related to security and privacy. These specific requirements, the complexities and the solutions put in place are reported and discussed in this paper, with special care regarding the satisfaction of GDPR and security aspects in the full IoT stack for smart cities and in some measure also for Industry 4.0.

The paper is organized as follows. In Section II, the major requirements identified by Snap4City from its challenge on Select4Cities and which are needed for secure and Smart City IoT platforms are listed. They have been mainly divided into functional and nonfunctional requirements, and a specific section is presented to identify security vulnerabilities to be avoided. In Section III, the related works are presented including an analysis of the most prominent solutions, focusing on aspects of security in the overall user data management workflow. Section IV presents the Snap4City Platform General Architecture addressing the aspects of security when the IoT solution is deployed on premise, on cloud and mixed. In the same section, the aspects of security are analyzed in the configurations on cloud, while the aspects of authentication and authorization as discussed in Section IV.D. In Section IV.C, a table reporting a matching of requirements vs the main Snap4City components and section ID is reported.

In Section V, more detailed architectural aspects regarding securing the IoT M2M (Machine 2 Machine) communications are presented, taking into account IoT Devices, IoT Edge Devices, IoT Brokers, and IoT Applications. Section VI discusses the remaining technical solutions to address

additional GDPR aspects that are indirectly connected to IoT and that are mandatory to complete the security shield of the infrastructure. In Section VII, the details regarding the validation of the Snap4City solution are reported. The validation included the verification of the criteria for GDPR compliance vs requirements, and the results obtained performing verification of vulnerability in Penetration Tests and the actions performed to solve the detected problematic areas. In Section VIII, the conclusions are drawn.

II. REQUIREMENT ANALYSIS AND PLATFORM COMPARISON

This section presents and discusses the major requirements that a platform for the Smart City IoT/IoE domain has to satisfy (in terms of privacy, security and GDPR). In some cases, the relationships with Industry 4.0 aspects are also presented as evidence to highlight the similarities. The overlap between Smart City and Industry 4.0 is relevant, since when a city has a water plant to monitor, energy consumption to control, etc., the problematic issues are those of Industry 4.0 plants. Since GDPR formalizes the aspects related to privacy, it is very difficult to separate GDPR-driven requirements from those about security. In the sequel, the requirements are presented in logical order from R1 to R18. In Section VI, the verification of GDPR compliance of the platform also addresses the related aspects of GDPR with respect to the following requirements. Therefore, the ideal Smart City IoT Platform has to establish the following.

R1. **support different kinds of IoT Brokers** and thus different IoT Devices and IoT Edge devices. ''Different'' in the sense that the Platform must be capable of supporting a set of IoT Brokers (on the cloud or remotely located on premise). They may provide different protocols and modalities to authenticate and establish secure connections with the Platform and Devices;

R2. **support IoT Discovery Abstraction** for IoT Devices. This means that the Platform must support the classification and search of IoT Devices, abstracting from their IoT Broker and protocol. Thus, in the activities of searching and subscription to IoT Devices, it has to be possible to identify them by searching for IoT Devices details such as: Device ID, sensors ID, geo-information (e.g., close to a GPS point, along a path, into an area, ...), sensor kind (e.g., temperature, humidity), nature (environment, mobility, energy, etc.), value unit (Celsius degree, micrograms per cube meter), protocol (NGSI, AMQP, COAP (Constrained Applications Protocol), etc.), Broker, etc. The result of the IoT Discovery process can be a set of IoT Devices or sensors that could be accessed / subscribed independently on their IoT Brokers and protocols;

R3. **guarantee authenticated connections** among IoT Devices, IoT Edge Devices, IoT Applications, storage on the cloud (so-called Data Shadow, as in AWS, IoT Azure), dashboards on secure channels. The IoT Devices can be in the field and on premise, while others can be on the cloud and mixed. The authenticated connections have to be established among each other, and in the best cases, by using mutual authentications. Moreover, as a fallback solution, by using less secure connection models, such as those based on keys and/or basic authentication (that can also be employed when the IoT devices are not compliant with most secure protocol approaches). Please note that some of the communications among entities are machine to machine, M2M, while others are human to machine, H2M;

R4. **inform users** about the security level at which the solution may work according to the level of security taken (it may depend on the kind of sensitive data managed);

R5. **support developers in managing security.** This means that the developers of IoT Applications should be supported in creating applications that exploit the security in a transparent manner as much as possible. For example, the developers of IoT Applications need to create connections with: Dashboards (for presenting data and collecting actions from users), storage (for accessing to historical data, or for saving additional data, results of some data analytics) and with IoT Brokers (for subscribing on the data drive, or sending/receiving messages), etc. IoT Applications may also invoke and implement Data Analytics processes exploiting a large amount of data storage, for example, by using machine learning approaches. Thus, the authentications to establish these M2M connections have to be automated. This means that the developers are not forced to use the credentials in the source code to establish authenticated connections (for example, with the IoT Brokers, Dashboards, Storage, etc.). Please note that this kind of security weakness happens in most of the IoT platforms in the state of the art, see for example, Azure IoT;

R6. **guarantee Secure Communications** in all kinds of connections involving IoT Devices, IoT Brokers, IoT Applications, Dashboards and storage. In some cases, the communications can be in PULL and/or PUSH. The secure communications have to be guaranteed by means of an authentication approach during which certificate and/or access tokens are used, and then activate SSL/TLS connections supported by mutual authentications in the best cases as described in the previous points. The communication in M2M does not have an allowance for some back-office authentications that are specific for each user;

R7. **support developers with open hardware and open source software** for implementing secure IoT Devices and IoT Edge Devices. In this context, most of the platforms use proprietary solutions/devices to guarantee the secure connections, see for example AWS IoT, Azure IoT suite, etc. This requirement is connected to R3 since the developers have to be enabled to create

their own secure devices to communicate with the rest of the IoT platform elements in a secure manner, with mutual authentications. The adoption of an open solution allows the adopters to develop their own devices that can be secure at the same level of native solutions provided by the platform provider;

R8. **support signed consent** to authorize the usage, access and management of the different **Data Types** of the Platform. The concept of Data Type is derived from GDPR and can be regarded as Data Category. In the context of IoT solutions, the Data Types can be: IoT Devices/Edges, IoT Applications, Dashboards, as well as data entities in the storage, time series, etc. According to GDPR, the authorization/delegation to manage personal data (Types) provided by a user to the IoT Platform management must be performed by using a **Signed Consent** and not via an informed consent as in the past. Any Data Type should have a specific signed consent, registered on the platform and the grant can be revoked any time by the user;

R9. **register and manage IoT Data Types** providing, receiving, managing, storing and retrieving Data Types, for personal data and related access control. According to GDPR, IoT Data Types must **start as private** for the user, and only after the creation, the user may decide to avail them as public or accessible to specific users by **delegation**. The delegation in access to Data Types has to be at level of: (i) IoT Data and data sets (group of IoT Devices); (ii) single IoT Device; (iii) single sensor/actuator value of an IoT Device, (iv) IoT Applications, (v) Dashboards, (vi) storage values, etc. For example, the owner of an IoT Device, collecting a number of personal health parameters (pressure, glucose, temperature, etc.), would be interested to grant access to a partner only about the glucose level (only for one of the sensors of the IoT Device), and keeping the others private. Therefore, only the owners or delegated users can access the data. To make a Data Type public must be the equivalent of publishing data anonymously;

R10. **manage IoT Data Type ownership** (permitting the change of ownership) and the access **delegations** according to the GDPR. In the delegation management, it must be possible to list them (**check** the grants provided) and to **revoke delegation**;

R11. **support roles, organization, and groups** to manage different kinds of user's categories/group. It has to be possible to provide access delegation at Data Types to user categories to avoid creating thousands of delegations every time a new user joins a group. For example, in the smart city context, certain sensors would be directly accessible for all the officers of the mobility area. Thus, when a member of a group is added/moved, the delegations to its entities do not need to be created/removed. This requirement is mainly derived from Living Lab aspects and for multitenancy support on the platform to allow the usage of multiple applications and groups on the same platform;

R12. **store personal data in an encrypted** way according to GDPR to prevent, in the event of breach, the identification of any specific individual compromised data;

R13. **provide at the users the** "***right to be forgotten***", according to GDPR. A user must be able to fully manage personal data and eventually requiring the **erasure** of them from the Platform. As a limiting case, the user must be entitled to request the deletion of all personal data including **user profile data**. Please note that the platform must permit access to collected data at law enforcement agencies for a limited time interval, for example, for two months, that is a type of waiting/investigation time;

R14. **support auditing** for each user, to monitor who has accessed their personal Data Types. The user has to access at the auditing data, obtaining details about the accesses, such as: when, where, how, and who accessed data; this feature is requested explicitly by the GDPR;

R15. **support data breach detection:** to implement automated methods to detect in a short time whenever some data and Data Type have been tampered or leaked; this feature is requested by the GDPR;

R16. **support accounting** in terms of collecting/computing metrics/indicators about resource consumption. For example, counting the number of IoT Devices/Data, IoT Applications, Dashboards, etc., that the user has registered, created, and requested. The assessment is the first step to enforce eventual limitations according to the user's role and/or for billing. In IoT platforms, it is very frequent that an accounting is performed on the basis of the number of messages exchanged (both H2M and M2M), the amount of data stored, the number of IoT Applications created, etc. The accounting feature is only marginally connected to the aspects of security and GDPR, while the insertion of limitations on the number of resources that each user may request/exploit to/from the platform is a form of security against denial of services and penetrations that in most cases, try to abuse the services for resource releases;

R17. **support data protection** (privacy and security) **by design** and by default, requiring controls built into products and services from the earliest stage;

R18. **ensure a level of security** by providing technical and organizational measures appropriate to the risks, including, but not limited to, pseudo-anonymization, confidentiality and integrity, and performing a periodic penetration test. This requirement has to be satisfied by the Platform since many stakeholders may provide several kinds of data with different kind of licenses that range from highly sensitive data to fully open data.

In addition to the security aspects, it is important to keep in mind a few more nonfunctional requirements, which are valid

for Smart City IoT Platforms and may also be for applications in the IoT smart home, Industry 4.0, etc. They are indirectly connected but relevant to privacy and security aspects. In fact, a modern IoT platform must:

- provide technical and organization measures to ensure **scalability** and cloud services for IoT Brokers, IoT Devices, and IoT Applications. This means that in most cases, they have to be managed as Virtual Machines or Containers in an architecture supporting vertical and horizontal scaling;
- provide technical and organization measures to ensure **availability, resilience, disaster recovery, periodic stress testing, pentest and workload**. This requirement has been identified in the context of Smart City IoT/IoE as a nonfunctional requirement related to the reliability and robustness of the platform to guarantee a high availability;
- provide support for Cloud-Fog data routing and local (on premise) IoT computation on the IoT Edge, on which the security must be guaranteed as well. This also means that the solution should be installable on the cloud and on premise, and mixed solutions may be viable as well;
- provide support for building Dashboards and data presentations, business intelligence, visual analytics with simple visual tools that can be used by nonprogrammers. Please note that the Dashboards are the front end of IoT Applications, and thus the connection to the rest of the IoT stack has to be performed by using authentications on secure connections, such as via HTTPS or Secure WS;
- provide support for registering and managing heterogeneous in/out sensors and actuators on IoT Devices and virtually on Dashboards such as: (i) virtual sensors as buttons, dimers, sliders, switches, etc. (which are elements in which the user acts creating data for the platform); (ii) virtual actuators for showing of real-time data such as: graphic representations of a bulb or of an engine, gauge, single content, speedometer, level bars, time trends, etc. (which are elements to show data on the user interface).

A. SPECIFIC REQUIREMENTS ON SECURITY

As described in Section VII, the verification of most of the above described requirements has to be performed by using test cases, user interface inspection, and for some of them by using code inspection. In addition, in order to verify the aspects of security, the usage of the so-called **Penetration** tests is quite diffuse. In the context of the Smart City IoT, the main vulnerabilities to be verified can be summarized as:

- **SQL Injection**. Injections flaws occur when unvalidated data are sent as part of a command or query to the interpreter. The infected data are executed by the interpreter by running unexpected commands or accessing data without permission.
- **Broken authentication**. This refers to situations when procedures for authentication and session management are implemented incorrectly. Thus, the attackers may obtain passwords, keys, session tokens, or exploit implementing weaknesses to assume the identity of other users.
- **Sensitive Data Exposure**. When web apps and APIs do not protect sensitive data, attackers can obtain the data for fraud, identity theft or other crimes.
- **XML External Entities (XXE).** Old or badly configured XML processors evaluate references to external entities in XML documents. External entities can be used to identify internal files using URIs, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
- **Broken Access Control**. This refers to bad controls about what authenticated users can do. Attackers can take advantage of the deficiencies to access unauthorized features and/or data, such as accessing users' accounts, changing permission and roles, etc.
- **Security Misconfiguration**. Good security requires proper configuration of tools and web servers. In most cases, the default configurations are not always secure. This also implies keeping software and libraries up to date.
- **Cross-Site Scripting**. This type of flaw occurs when a web application receives data from untrusted sources and sends them to a browser without proper validation and/or ''escaping''. This kind of attack may allow running malicious scripts on targets' browsers; such scripts can hijack the user's session, deface the website or redirect the user to a malicious site.
- **Insecure Deserialization.** Unsafe deserialization may lead to code execution in a remote manner. In addition, these kinds of attacks can be used to perform privilege escalation attacks.
- **Using Components with Known Vulnerabilities.** Components, such as libraries, frameworks, and other software modules, work with the same privileges as the application. Once identifying the version, it could be possible to exploit their well-known vulnerability to perform the attacks.
- **Insufficient Logging & Monitoring.** Insufficient logging and monitoring, associated with a missing or ineffective integration with incident response, allow attackers to further attack systems, maintain persistence, rotate across multiple systems, and tamper, extract, or destroy data. Typically, the attacks are detected several days after the effective attack.

III. RELATED WORKS

A Smart City IoT Platform should be usually accessible and used by a number of developers and operators and by final users that could have very low (or maybe nothing at all) knowledge and understanding of computer programming and communication protocols. The hiding of the IoT technical

aspects and complexity, and at the same time supporting GDPR and security of personal/processed data is a very challenging task. There are some technical specific solutions frequently adopted by security practitioners that can drive the design of many basic aspects of security that people can use trustfully. On the other hand, there is still an open issue of how to create a full system that is satisfactory for all the mentioned security and privacy requirements in a user-friendly transparent manner for the different kinds of users.

In the literature and markets, several IoT platforms have already been proposed to the audience, even if just few of them have been tested on a large scale and on wide international geographical areas [14]. There are several interesting products that have been deployed for industrial scenarios and some of them that clearly still need vast improvements, mainly in data pipeline management and on the usability side. The platforms that support more flexibility usually became more difficult to use/configure, even by a user that is knowledgeable about IoT concepts and computer programming.

In the context of related work about IoT security, it is worthwhile to take into account the technological approaches that have been used to support the privacy and protection of data in distributed environments (IoT deployment). Quite comprehensive surveys about the different enabling technologies can be found in [15] and [16].

In **Table 1**, a comparison among Smart City IoT Platforms based on the identified requirements including GDPR is reported. The requirements listed in the table are a synthesis of those listed and derived from Select4Cities and ENOL as described in the Introduction. Table 1 reports the assessment with respect to the Snap4City platform that is presented in this paper. In the following paragraphs, we are presenting a review of the main technologies, research-oriented solutions and industrial platforms in the context of the Smart City IoT. For the industrial platforms, we have considered AWS IoT by Amazon [43], MS Azure IoT [44], and Google IoT [45], since we think that only these large platforms can be comparable in terms of security/privacy requirements coverage with respect to the Snap4City solution and architecture. The Snap4City general architecture has also been compared with other solutions in [17].

Among specific IoT security solutions for the ARM (Advanced RISC Machine) microcontroller, the *ARM mbed* IoT has the advantage [48] over the other platforms to provide an operating system for development, while the support for user privacy and devices' healthiness automatic check is overlooked. Calvin IoT platform by Ericsson [49] presents a five-layer architecture deployed on top of devices and has been substituted with the new Kappa solution. The HomeKit solution by Apple mainly focuses on features related to the smart-home scenario [50]. All of them, in addition to Calvin, offer authentication of smart devices via the X.509 certificate and an internal ad hoc solution for authorization. The TLS/SSL (Secure Sockets Layer) channel is always involved in overall communication. Minor solutions are the AirVantage stack which is an ecosystem of IoT projects provided by the Eclipse consortium [51], and SmartThings from Samsung [52] that focuses on the home-connected scenario and permits authentication via OAuth2.0 protocols over SSL/TLS channels and authorization via an ad hoc capability-model.

Many techniques for secure communications have received an increment of interest in the last period due to the direct application in the IoT context [18]. Some of them are directly employed in the definition on privacy models for reciprocal trust. For example, Data Tagging permits associating labels to the dataflow in order to grant trusted entities access [19] mainly in a social-bookmarking context [20] or in platforms for photo sharing [21]. Zero Knowledge Proof allows managing user's privacy between two entities without the need of the prover to reveal any information to the verifier [22], [23]. Elliptic curve-based zero knowledge proofs has been used in devices with strong constrained resources to establish trust between two parties [24]. Group-key management technologies enable cipher support in a group communication, and thus also in distributed networks [25] with novel approaches for the IoT scenario [26]. Finally, the K-anonymity privacy model permits disclosing entity-specific information such that the released data cannot be reliably linked back [27], [28].

Other solutions for IoT security have been based on the Blockchain paradigm. Some steps toward a complete IoT platform based on Blockchain concepts have been made and look promising. For example, the usage of Blockchain for the management of authentication and authorization via Smart Contracts [29] is such an application. It relies on an external (central) repository or a key-management system and not on a distributed and decentralized approach [30]. In addition, some problems related to the scalability have also been detected, regarding the management of policies for authorizations and the risks of race attacks [31]. In this paper, we are not addressing IoT Blockchain solutions since, for the moment, they do not cover the entire security stack and do not satisfy GDPR.

Moreover, there are a number of proposals that aim at increasing the control of privacy in the IoT context. These solutions can contribute to end-to-end communications of the platforms. The most relevant are the obfuscation systems that try to provide the anonymization of location-based services through spatial and temporal cloaking [32], [33] or over noisy channels [34]. Other specific methods about privacy and protection have been applied in medical data field [35], using pseudo-anonymization techniques that focus on decupling data from the owner to enable a secondary use of data with respect to data privacy [36] or to enable trusted third parties via privacy-enhancing techniques (PETs) in the context of data collection [37]. Some other researchers focused their attention on the specific problem of enabling trust between different components in a distributed IoT architecture [38], [39].

Among the most interesting widespread solutions, we can surely place Amazon AWS IoT (Amazon Web Services) [43]. This platform lets smart devices connect and interact each other and with the AWS Cloud. AWS IoT makes wide

TABLE 1. Comparison of the mentioned IoT Platform on the basis of the main requirements identified. The table is divided in two parts: Open source and proprietary solutions. When: G is reported means that the requirements have been mainly derived from the GDPR; "( )" are used to mark partial (The assessment refers features accessible at the date of submission.).

	IoT Discovery Abstraction	Authentication, Authorization	Security end-2-end, secure on IoT and Dashboards	Open HW and Open SW	Integrated Community management	Data Types: IoT Devices, IoT App, Dashboard, Data	Data Type: Publish/share, Delegation, Consent and change	Data Type: Download and Delete	Auditing on Data Type Access	Open Source end-to-end	Scalability IoT	Visual Programming end-to-end applications	Advanced Smart City API, MicroServices	Multi Domain Semantic Platform	Standard based Modules and IoT, Open Devices	Resource Sharing	Data Analytics integrated	Dashboard H24/7, protected connection	Multi-protocol on IoT
		G				G	G	G	G										
Snap4City	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
KAA [53]	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	N	Y	N	(Y)	N	N	Y	Y
Thingsboard [55]	Y	Y	Y	Y	N	Y	N	Y	Y	Y	Y1	N	N	N	N	N	N	Y	MQTT,coap, http
IoT eclipse.org [56]	N	N	N	(Y)	N	Y	N	N	N	Y	Y	N	N	N	Y	N	N	N	Y
IoT IGNITE [57]	N	Y	N	Y	N	Y	N	Y	Y	Y	Y1	Y	N	N	N	N	N	Y	MQTT
FIWARE [47]	N	Y	N	Y	N	N	N	Y	N	Y	(Y)	(N)	Y	N	Y	N	N	Y	Y
ARM mbed IoT [48]	Y	Y	Y	Y	Y	N	(N)	N	Y	Y	Y	N	N	N	Y	N	N	Y	Limited
Airvantage [51]	Y	Y	Y	Y	N	Y	N	Y	Y	Y	Y	N	N	N	N	N	N	Y	MQTT, HTTP
AWS [43]	Y	Y	Y	Y	N	Y	(N)	Y	Y	N	Y	N	N	N	Y	Y	(Y)	Y	Limited
Azure IoT [44]	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	Y	Y	(Y)	Y	Limited
PTC ThingWorkx [59]	N	Y	Y	Y	Y	Y	N	N	Y	N	Y	Y	N	N	Y	N	N	Y	Y
Bosch IoT Suite [58]	Y	Y	Y	Y	Y	(Y)	(N)	Y	Y	N	Y	Y	Y	N	Y	N	Y	Y	Y
CISCO Jasper [55]	Y	Y	Y	Y	N	(Y)	(N)	N	Y	N	Y	N	N	N	N	--	(Y)	Y	N1
Siemens MindSphere [60]	Y	Y	Y	(Y)	N	Y	(N)	Y	Y	N	Y	Y	N	N	Y	N	Y	Y	Y
Carriots [54]	Y	Y	Y	(Y)	N	Y	N	N	Y	N	Y	N	N	N	--	N	N	Y	MQTT
Google IoT [45]	Y	Y	Y	Y	Y	Y	N	Y	Y	N	Y	N	N	N	N	N	(Y)	(Y)	MQTT, HTTP
Homekit Apple [50]	Y	Y	Y	Y	N	Y	N	N	Y	N	(Y)	N	N	N	N	Y	N	Y	Limited
Smarthing Samsung [52]	Y	Y	Y	Y	Y	Y	(Y)	Y	Y	N	(Y)	N	N	N	N	N	N	Y	Limited

use of proprietary services such as Amazon Dynamo DB (database), Amazon S3 (simple storage service), Amazon Machine Learning and others. It consists of a cloud solution that provides a data shadowing system to support intermitting connectivity via the communication protocols MQTT, HTTP1.1 (Hypertext Transfer Protocol) and WebSocket. A Registry Unit is used to collect any information about the IoT Devices and to track their behavior. The main supported security features are:

- authentication at the device level via X.509 certificate and via AWS authentication;
- authorization and access control by mapping the X.509 certificate to a policies-based system through a set of rules processed by a so-called Rules Engine;
- secure communication of traffic is ensured to be made over SSL/TLS protocol to ensure the confidentiality of the application protocols, only for their supported protocol and devices.

The AWS IoT ecosystem is composed of a complex stack of Amazon's services that enable the developers to define the business logic via programming the data flow among IoT Devices' toward their visualization in proprietary dashboards. The variety of tools for IoT programming is wide and relies on several AWS components (*AWS Batch jobs, AWS Step functions, Amazon MQ* (Message Queue, with an IoT Broker as Active MQ) and many more). Moreover, the IoT Devices can trigger *AWS Lambda Functions* that may be written in Java, Python and C#. The Lambda Functions are used to implement business logic or to trigger actions in the AWS Cloud or on

premises with some limitations. The data visualization is rich and flexible, and it can be exploited for data inspection and reporting. The *AWS IoT Device Defender* module can be configured to create alerts on a data breach on a singular device. Accounting on the activity of the platform is rich and detailed: it makes strong use of the *IAM users* (Identify and Access Management) module to specify groups and roles and to cluster/aggregate data and information. Limitations of this platform are discussed in the sequel.

Another well-known platform is the **Microsoft Azure IoT Suite**[44]. It enables the usage of smart devices to the *Azure cloud* via the modules called *Cloud Gateway* and *IoT Hub.* It manages the identity and authentication of registered devices with an identity registry and natively supports communication protocols such as AMQP (Advanced Message Queuing Protocol), MQTT and HTTP. On top of the communication layer, there is the *IoT Solution Backend layer* that presents a set of Azure services (as machine learning and analytics) and the *Presentation Layer* that is involved in the visual presentation of the data. The main security features of the Azure solution are:

- authentication is mutual using the SAS Token (Shared Access Signature) and X.509 certificate;
- authorization and access control are realized via *Azure Active Directory* using a policy-based model to support near-instant revocation;
- secure communication using SSL/TLS protocol; the identity registry implements the secure storage manager.

Azure IoT provides a development environment to build business logic. The integration toward mobile applications is made by using the Xamarin platform in C# and .NET programming languages. In addition to the mobile environment, it is possible to also write applications in Java, Node.js and Python. Security is provided by the IoT Hub. For the creation of complex dashboards to visualize the data, the developers can use the Microsoft *Business intelligence* tool and a detailed role-based user access control is available to provide a rich description of the user characteristics and rights (Owners, Contribution, Reader, User Access Administrator).

Finally, the **Google Cloud IoT** is a set of tools to connect and process smart devices' data in the cloud and at the edge [45]. It is composed of cloud services that permit native support of MQTT, HTTP and AMQP protocols over TLS/SSL via a two-fold subsystem called Brillo and Weave. The offered security features are:

- authentication via Oauth2.0 (Open Authentication) protocol along with a digital certificate;
- authorization and access control involve the SELinux module to manage the access control security policies. The enforcement is made above the right to read, execute and write for any users or group of users;
- secure communication provided through SSL/TLS protocol.

Google IoT platform makes strong use of the access token (to enable authenticated access to the data) in the form of JWT (JSON (JavaScript Object Notation) Web Tokens), to enable access between the components. On the other hand, it is not direct for the IoT developers to inject credentials in the Application Logic to exploit them. Tokens are created when a new IoT Device is registered, and credentials may need to be specified in clear text in the application code. The documentation for the platform is not of immediate usage for experts. A detailed set of APIs (Application Program Interfaces) is provided, and only simple cases are presented that require knowledge of Apache Beam. Several programming languages can be used to program the data flows from the device to dashboards, such as Java, Node.js, Python, Go, Ruby, PHP, C#.

Amazon AWS IoT and Microsoft Azure IoT Suite solutions strongly rely on proprietary and closed sources for the user identification and access control enforcement (AWS IAM, Azure Active Directory, respectively). In meantime, the Google IoT platform supports integration with the open OpenID Connect protocol. All these platforms have strong limitations on the supported number of protocols (mainly just HTTP and MQTT), and on the number and IoT Brokers types that are supported (very limited support on managing multiple brokers). For the security connection, the modalities in which the transport level security is guaranteed are very similar: TLS/SSL and X.509 certificate. In most cases, from their proprietary certified IoT Devices and toward their IoT Brokers only. In most cases, when the user passes to develop IoT Applications and attempts to exploit communication with custom devices, the security is not natively guaranteed. Moreover, from the documentation, it is not evident how to implement custom scenarios and secure devices. In these cases, the definition and generation of security credentials (access tokens) is not supported at runtime, and it is used only when the IoT Device is registered to the platforms. It is difficult to find specific ways to enforce the access control in real time. Specific tools to label and classify the type of data that flow in the platform by default were not found in the analyzed solutions and none of them permits specifying the way that data authentication is provided. In those platforms, a common approach is taken for programming the data flow: programming languages can be used to implement a public interface so that the platform can use the written source code to manage the data. None of them provide a visual editor for creating IoT Application flows that can be used without any software programming skills. In addition, these solutions often require mandatory cloud services and are not suitable for deploying on an autonomous on-premise scenario. As a limiting case, they may accept having some processes on the IoT Edge on premise, with reduced capabilities connected with the cloud solution.

It is important to notice that, in the context of IoT, different solutions may need to satisfy different use case scenarios. For example, in the health-care domain the personal data must be protected in a stronger way and real-time requirements must be strictly satisfied with respect to what could be needed in the entertainment domain. A complete study of the

classification of different scenarios and their required security requirements has been made in [40]. For example, among the considerations are clarifying differences among aspects such as object security, end-to-end security, cyber-physical-social security, hierarchical security, lightweight security and defense. For this reason, in the next section, we describe the specific requirements on privacy and security that have been identified in the context of the Select4Cities Smart City IoT/IoE challenge, with some additions that have been applied to address the aspects of the Industry 4.0 domain that are needed in city plants/installations with the Snap4City architecture design.

Other solutions such as Kaa [53], CISCO [55], CarrIoTs [54], and Thingsboard [55], support IoT networking, without visual programming, semantic management, and limited control and integration with smart city data. A relevant lack in the IoT management of devices for IoT Discovery is registered to the platform as IoT Eclipse.org [56] and IoT Ignite [57]. Most of them lack semantic search support, such as Bosh IoT [58] and PCT ThingWorkx platforms. For these last two platforms, the effective capabilities in support of GDPR are not very clear, especially for supporting multiple Data Types as those needed in Smart Cities, which are more suitable for industry applications.

IV. SNAP4CITY PLATFORM ARCHITECTURE

The Snap4City architecture has been designed and implemented to satisfy the above described requirements that have been introduced in principles described in this section. Thus, Snap4City allows being deployed on the cloud and on premise, as a well as in mixed scenarios, and supports IoT Edge Devices with IoT Applications, and IoT Applications on the cloud as well. The Snap4City architecture is flexible enough to satisfy the above requirements which are valid for a wide range of IoT domains with local computation on premise, on cloud and mixed. In Snap4City, a set of tools can guarantee the privacy and protection of the data managed by the system respecting the GDPR, requirements of ENOLL and those of Select4Cities. Snap4City was the winning solution of the Select4Cities PCP and challenge.

The Snap4City.org is the cloud-based backbone of the solution. It acts as the boilerplate where different city organizations and City Operators configure, via a set of graphical tools, the processes for data aggregation, data analytics, and data rendering via dashboards. The data generated by IoT Applications can be injected back in the storage and in the Knowledge Base by making them accessible for MicroServices and Dashboards, and thus in connection with the IoT installations in the field. Therefore, in the following paragraphs, the different scenarios are presented, keeping in mind that all of them persist at the same time in the actual deployments.

A. GENERAL ARCHITECTURE

The Snap4City architecture highlighting the security aspects is reported in Figure 2. From the upper left part, the system permits accessing the platform at the users with a set of devices/terminals via a Web User Interface and/or mobile applications. The users may access the platform functionalities in an authenticated manner. The user's identity is verified and propagated by the Single Sign-On (SSO) Login module toward any Snap4City module that needs to access some user's data. The SSO allows performing a centralized authentication management to all the modules and tools of the infrastructure. The users' personal data (mandatory and optional information about the user) are kept in a separate storage called User Registry.

The platform is interfaced to the real world, in which a multitude of IoT Devices are located and supported (hosting several sensors and actuators). A simple IoT Device can be a microcontroller enriched with sensors/actuators capable of send/receive messages to/from a specific gateway (Lora, SigFox, OneM2M, etc.) or directly with some IoT Brokers that can be external or internal to the Snap4City platform.

Since most of the IoT Brokers do not support native authentication and channel protection (such as IoT Orion Broker of Fi-Ware), they can be protected by an **IoT Firewall** (a shield) that allows performing only authenticated connections on the basis of the SSO and thus allows or denies access according to the requests. The IoT Firewall performs the access control establishing a mutual authentication and a secure connection over HTTPS, exploiting OMA NGSI Ver.1 and Ver.2 protocols (Open Mobile Appliance, Next Generation Service Interfaces). The IoT Firewall supports other authentication mechanisms such as those based on: (i) access token, (ii) a couple of keys (as in SigFox), and (iii) basic authentication over HTTPS. It can be easily adopted for wrapping other protocols and IoT Brokers.

B. THE ROLE OF THE IoT EDGE ON PREMISE VS SECURITY

IoT Edge Devices are a powerful class of IoT Devices that is capable of (i) managing sensors/actuators IoT Devices, send/receive messages to/from some IoT Gateway, and (ii) executing IoT Applications (processes with some local logic, for example, for data aggregation, data mining, machine learning, artificial intelligence, AI, etc.). In Snap4City, IoT Edge Devices can be based on Raspberry pi, Linux, Windows, Docker and/or Android. Thus, powerful computer-based devices can be delegated to play the role of IoT Edge and not only embedded systems. IoT Edge Devices may have capabilities to connect on multiple communication channels including local wireless networks. These kinds of devices may play the role of a message router toward some IoT Brokers (for example from ModBUS to Snap4City). More complex *handheld* personal computers (e.g., Android on smartphones or similar devices) can also be used with the Snap4City platform. They are IoT Edge Devices with the embedded strong network capabilities toward the Internet via 4G, 5G connections to directly communicate with the infrastructure (for example, to access the historical data for machine learning). In all these configurations, the solution has to guarantee secure connections and authentications.

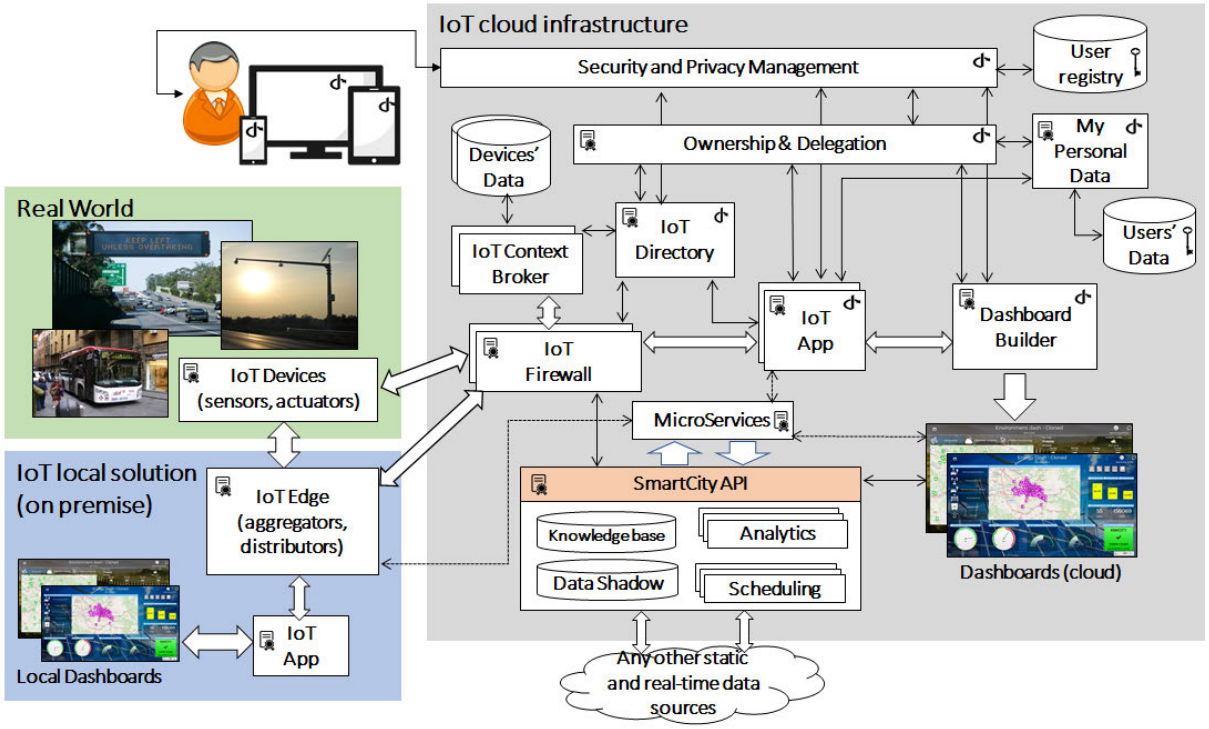

FIGURE 2. Snap4City architecture, main security aspects.

As a remark, any supported IoT Device can communicate directly to the platform via an Internet communication reaching some IoT Broker on the cloud. Otherwise, they can be configured to be connected with Snap4City using an aggregator/distributor device (**IoT Edge**) connected, for example, via a local wireless network interface toward an access point (in IEEE 802.x), or other private/patented technologies (LoRa, SigFox, RS485, ModBUS, BLE (Bluetooth low energy)), thus playing the role of an IoT Gateway. The IoT Edge in turn would be capable to communicate with the platform on premise or on the cloud via the Internet.

Moreover, **Snap4City IoT Edge Devices** may have IoT Applications exploiting Snap4City services on the cloud. IoT Applications in Snap4City are node.js processes that can be formalized by using the visual syntax and model of Node-RED [46]. The exploitation of Snap4City services is performed by installing the Snap4City suites of **MicroServices** (see **Figure 3**), directly from the Node-RED Library https://flows.nodered.org/?term=snap4city of the JS Foundation.The libraries allow easily invoking MicroServices on the cloud using them as nodes and they establish secure connections. They include services such as: IoT Brokers, IoT Discovery, Big Data storage, Data Shadow, Data Analytic (artificial intelligence and machine learning), External Services and Dashboards, plus a large number of specific MicroServices for Smart City services, and Industry 4.0. Examples of Smart City MicroServices are: search for paths of buses and time schedules, parking status, routing, parking predictions, anomaly detection, traffic flow reconstruction, environmental data values at any point of the city, etc. [65]. The Snap4City MicroServices exploit the Advanced **Smart City APIs** [41], which in turn are largely based on the semantic engine of the Km4City **Knowledge Base** (KB) for smart city data and service management.

IoT Edge Devices may need to provide results via local Dashboards (accessible on an intranet, on premise, for example, by using Node-RED classic Dashboards). More sophisticated Dashboards can be produced exploiting Snap4City cloud data and capabilities, via secure communications, such on WS (Web Socket), to guarantee the real-time stream with virtual sensors/actuators on Dashboards. They are also called ***Virtual*IoT Devices.** Thus, a set of predefined sensors and actuators, such as buttons, dimers/knobs/sliders, lights, pumps, fans, traffic light, etc., and custom widgets, can be created.

C. IoT CLOUD SCENARIO VS SECURITY, GENERAL VIEW

When an IoT solution on the cloud is chosen (as shown in **Figure 4),** the data flows from/to the IoT infrastructure via a set of **Context Brokers** (Mosquito MQTT, Fi-Ware Orion Broker NGSI, etc.) without the usage of IoT Edges. The IoT data streams arriving on the cloud are automatically saved on Data Shadow for creating the historical IoT data.

The data collected in the Data Shadow is made available to any process of the Snap4City solution, mainly to the IoT Applications deployed on cloud via the MicroServices. The IoT Applications (based on Node-RED plus the Snap4City library of nodes/MicroServices for the Smart City)

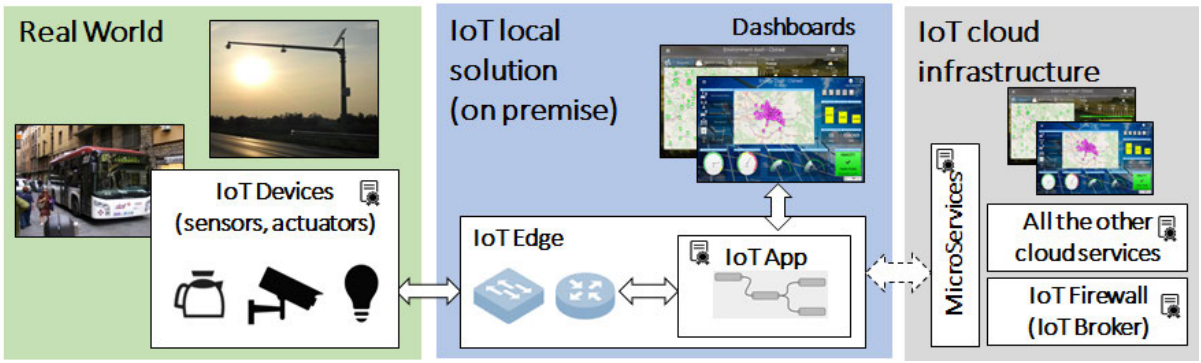

FIGURE 3. Snap4City configuration for local solution (with not mandatory use of IoT cloud infrastructure).

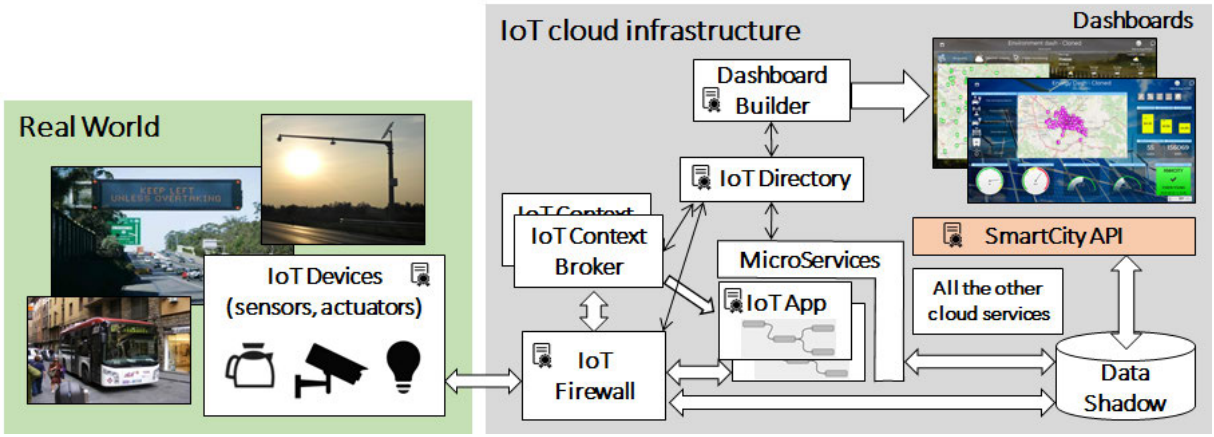

FIGURE 4. Snap4City configuration for a complete cloud solution.

allow the user to design, in a visual manner, personal data flow IoT Applications for data transformation, management, and data processing. The MicroServices are the means to invoke Data Analytic processes on the cloud to access storage, and domain specific services such as those described in the previous section. The IoT Applications can integrate visual elements of Dashboards (called widgets), which can be improved and integrated with smart city data by using the **Dashboard Builder [7]**. The Dashboard MicroServices (Snap4City nodes of Node-RED flows) allow users to compose in visual and intuitive manners the data presentations as Dashboard Widgets for full user interaction and include virtual sensors and actuators, custom widgets, exploiting smart city API, maps, etc. The Security of the Smart City API is taken for granted in this paper, they are Rest Call as described and discussed in [17].

Since the IoT Platform needs to connect multiple IoT Brokers and Devices, an **IoT Directory** listing them and providing general abstraction services is needed. From the security point of view, the IoT Directory must manage the IoT Device's registry, including references to the credentials that IoT Devices have to provide at the **IoT Firewall** to establish a secure connection on the IoT Platform. In this regard, different levels/models of authentication are available (key credentials, certificate credentials) and are specified whenever a new device is registered by the user in the Snap4City ecosystem. The IoT Device messages can also flow directly to the IoT Applications, where, in the form of aggregated data, they can be part of data analytics and data refinement. This means that the IoT Application has to be capable of subscribing to the IoT Device, via the IoT Broker to receive data in push.

Any IoT entity, such as data, IoT Devices (with their sensors and actuators), IoT Applications and Dashboards, whenever they are instantiated/created (on the cloud or on premise), have to be associated at their creation with a user (the owner).

The IoT entity owner is the only one authorized to manipulate those entities, and thus entitled to provide delegation in access to those elements. The rights of the entities (both ownership and delegation) are managed by the **Ownership** module which can be queried only in an authenticated manner. Dedicated tools for the entity and Data Type are used to grant/deny access to a user/group/ organization. For example, the IoT Directory is the only tool for changing the ownership and creating delegation on IoT Devices. The users can also

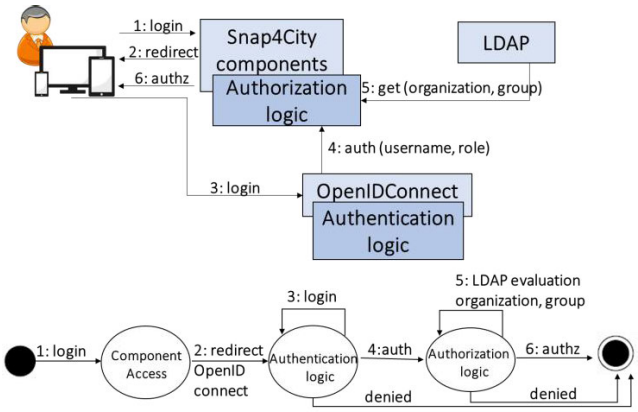

FIGURE 5. **Snap4City communication and state-transition diagram about the management of users' authentication/ authorization.**

make public their own IoT entities, which are implemented by creating a "*Delegation to Anyone*" in an anonymous form according to GDPR.

The ownership and delegation data, in addition to any other personal data, are put in a private and secure storage managed by the **MyPersonalData** module, in conformance to the GDPR. MyPersonalData are labeled using a classification based on motivation, variable name, variable value, and variable unit. Once defined, the developers can exploit the MyPersonalData safe storage via MicroServices or APIs and may also render these data on Dashboard Widgets. In addition to the personal data, the so-called **MyKPI** (Key Performance Indicator manager) is also available. They can be used to manage time series with variable GPS locations. They are typically used for storing mobile sensors, values collected from mobile phone movements (personal paths), on board unit of vehicle data, etc.

D. SECURITY OF USER/MACHINE ACCESS AND AUDITING

Please note that in the IoT stack (as depicted in Figure 1), a number of entities associated with the users may need to preserve privacy and security, such as: IoT Devices, IoT Applications, IoT Edges, Dashboards, and data storage. Therefore, before entering into the IoT aspects of security, it is mandatory to describe the mechanism for user/machine authentication and authorization.

The Snap4City solution uses several mechanisms to provide the **authentication and authorization** enforcement for users/entities to access the platform resources (see **Figure 5**). The User Registry is partially managed by a **distributed directory information service** (based on LDAP, Lightweight Directory Access Protocol) and partially maintained by a CRM (customer-relationship management) based on Drupal (which are maintained synchronized). The LDAP module is reachable only via a private internal subnetwork such that it is more easily protected from attacks. On the CRM, the users can review and update their information and eventually request to completely remove their account data according to the GDPR. Mandatory information is saved in LDAP and includes: usernames, hashed version of the user's password, email, roles and organizations/groups affiliations. The username is the primary key used to identify the users in the overall Snap4City platform. The roles (Manager, Area-Manager, ToolAdmin and RootAdmin) are used for users' classification regarding their trust level in the framework and to grant access to Snap4City functionalities (e.g., enable different views of the interface, permit more or less data investigation). Moreover, the public/anonymous users are also authorized to play with some tools, even if with limited features. The organization/group information is used as a multitenancy key to organize the users in terms of location or purpose and affiliation (for organization, e.g., Firenze, Helsinki, Antwerp, DISIT). For each organization, a number of groups can be set up and each organization may define its own names for them (for example: Developers, ICT officials, third-party developers, citizens, decision makers).

The user authentication is enforced by using a **Single Sign-On (SSO)** module with **Identity Management.** It exploits the **OpenIDConnect** protocol to provide a users' authentication system for all Snap4City modules (which include the CRM module based on Drupal, and Keycloak for the OpenIDConnect). Since OpenIDConnect is based on OAuth 2.0, the latter is used for the authentication enriched with the user's identification part. According to the flow numbers reported in **Figure 5**, every time a user tries to access a resource exposed by a Snap4City module (1), the request is forwarded to the SSO Server to verify if the user is already logged (2). In the case in which the user is not already logged, the SSO Server requires the user to specify user's credentials (3). Whenever there is a match to the credentials stored in the LDAP users' register, the user is authenticated (4). Then, it has to be *eventually* authorized (6) to access the resource exposed by the Snap4City module (depending on ad hoc additional enforcement provided by the modules (5)). The user's role specified in LDAP is mapped in the SSO rules' registry. Thus, the system administrator can specify, in particular, the access rule (5) to enable or deny access (specification of grants) to a specific user's role to specific Snap4City modules. During the authentication, Keycloak passes to the contacted Snap4City modules the usernames and their roles. Whenever the user is granted to access a specific module, the module can implement another finer ad hoc authorization's rule to eventually authorize/deny the requested resource (6). The module could also contact the LDAP to retrieve the organizations and groups the user belongs to.

Any communication between a couple of Snap4City modules is made on top of the **SSL/TLS protocol.** Thus, transmission is kept confidential and a system of temporary shared secrets, represented in the form of an access token, JWT, that permits the SSO system to work among the different modules is used. As described above, when the user accesses in an *interactive* manner, if a JWT is not present or not valid (elapsed, tampered or not correctly digitally signed), it is

redirected to the login Web page (3). When a M2M connection needs to be established; for example, an entity (such as an IoT Edge Device or an IoT Application) needs to access to the Snap4City MicroService module, a formal user still needs to specify credentials, at least for the first time. Then, an offline-access token is released (refresh token), and the machine does not need to request the user to login anymore and can use the refresh token to request a normal access token as specified above. The refresh offline token has a long lifetime, so that human intervention is limited, and it is maintained in a safe. Once the authentication and identification of the user is successfully completed (a valid access token is returned), the Snap4City modules are contacted attaching the JWT as their credentials. They are used to verify if the user/machine has enough rights to access the requested data/resource.

The OpenIDConnect protocol enables the identification of entry points and of communications among different Snap4City modules to authenticate the users/machines. This configuration permits realizing a good user experience; and at the same time, setting up mechanisms for detailed auditing of user accesses (see **Figure 6**), which is a demanded requirement for infrastructure monitoring. Moreover, the secrets that are used by the different Snap4City modules to access the SSO Server for user's authentication are specific for any module; thus, in the case of leaking or malicious intrusions, the problem can be isolated, and there is no need of a complete system reconfiguration.

FIGURE 6. Auditing of module's activities (some text has been intentionally obfuscated).

E. ARCHITECTURAL CONSIDERATIONS

As a summary, in Table 1, the main requirements and their mapping on architectural components are reported. The mapping also provides evidence regarding which are the main sections and subsections in which those aspects are addressed in the paper. Please note that a systematic visualization of all the aspects would take too much space; therefore, the paper has been optimized to show only the most relevant and innovative aspects and presents them by scenarios. Moreover, the Snap4City platform is compliant with several protocols, documented and has several test cases https://www.snap4city.org/283. In addition, all the source code is on GitHub and is 100% open source including the security aspects from code of IoT devices to that of Dashboards.

V. IoT M2M SECURE CONNECTIONS

In this section, the architectural mechanisms for establishing secure M2M communications are discussed and presented. In detail, the security aspects addressed are those related to: IoT Device communications on premise and on the cloud; IoT Applications communications with respect to devices on premise and on the cloud; and communication of IoT Devices and IoT Applications with respect to Dashboards.

A. IoT NETWORK, DEVICES VS SECURITY

Any IoT Device has to be registered in the IoT Directory to enable the communication from/to an IoT Device with the infrastructure (stating also the security model adopted, if any, and obtaining an IoT Broker). For IoT Device registration, the user has to specify a unique identifier, type/manufacturer, and location on a map. The IoT Broker can be internal, and thus it can be chosen/assigned on the basis of the protocol. As an alternative, the user may refer to an external IoT Broker, which can be registered on the platform as well. When the number of devices to be registered is massive, the user can use a procedure to automatically register a large set of devices starting from a detailed CSV (comma separated value format) file containing all the needed information (i.e., so-called Bulk Registration). According to the requirements, several different authentication schemas for different kinds of IoT Devices and protection levels have to be supported and have been implemented as described in the following. In addition, all the IoT Devices at their registration start as private in terms of the user that registered them, that is: private as default according to GDPR.

*In many **proprietary solutions**, the interaction with the IoT Devices and thus also their authentication is completely demanded to the gateway/server infrastructure.* In those cases, the IoT Devices exchange messages/data only with their own infrastructure servers (e.g., for SigFox configuration, the SigFox server provides two keys: K1, K2, as credentials). As a consequence, in Snap4City, as in other interoperable IoT Platforms, it is possible to obtain data access in pull or receive them in push (after subscription). Once a data message arrives in the Snap4City platform, it is associated with the owner of the IoT Device disregarding if the message has been pulled or received in push. In the case

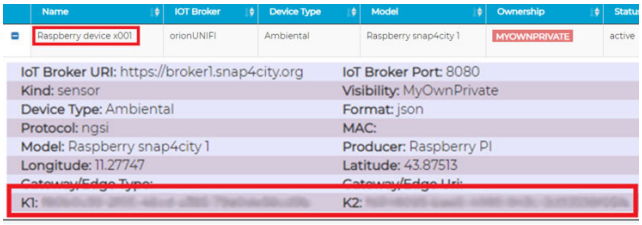

FIGURE 7. Details of a registered IoT devices.

in which an IoT Application is developed for data collection, the application itself (for example, a coded program) is forced to include in the code body the credentials and this could be regarded as a breach of security since one accessing the IoT Application code would also access the credentials. In Snap4City, in order to avoid this weakness, the MyPersonalData safe storage is used for automatically storing of the IoT Device credentials which are used for subscription (push) or for each server access in pull. Thus, IoT Device credentials are used only at the execution time on the basis of the passed refresh token and are not present in the code. The whole mechanism is automated if the user registers on the platform some SigFox or other proprietary authorization schemas.

*For **Snap4City Open IoT Devices** (devices that are provided as Open Hardware/Software*, or derived solutions), the platform can rely on several communication protocols, such as NGSI, MQTT, COAP and AMQ, and provides a Snap4City protection system. If NGSI is chosen, the Orion Context Broker of Fi-Ware is used with the IoT Firewall. Examples of Open IoT Devices of Snap4City are: IoT Button ESP32, Arduino based IoT Device, Raspberry pi IoT Edge (open SW), etc., of which the source code can be downloaded from the Snap4City Portal.

A **simple authentication modality** makes use of a pair of **keys** (K1, K2) as credentials. In this case, they are automatically generated by the Snap4City when the IoT Device is registered in the IoT Directory and are provided to the device owner for setting them into the IoT Device (see **Figure**7). When an IoT Device needs to communicate in push with the Snap4City framework (e.g., a value of a sensor is updated), it has to establish the communication using K1, K2 credentials and its own IoT Device identifier.

Moreover, in Snap4City, according to GDPR, an additional set of keys is generated when the **IoT Device owner delegates** in access (read-only) data to another user, or group of users/organizations. The delegation can be performed at level of single sensor or for the whole IoT Device. When the delegation is removed, the K1 and K2 are erased and will no longer be valid. When an IoT Device is declared "*Public*", an *Anonymous* delegation is generated; in this later case, the K1, K2 can be omitted, and any access operations will be always permitted. In any case, only the owner may modify the internal settings/values of the device.

*In Snap4City, a**higher level of security can be chosen by the IoT Device owner*** in terms of the security barriers' authentication solution, which is based on the **X.509 certificate and mutual authentication**. This technique involves the exchange of a signed digital certificate, based on a private-public key cryptography solution. The Snap4City IoT Directory acts as a security and identity unit using a self-signed Certification Authority. The exchanged certificates are SSL/TLS-based, to ensure secure authentication: the HTTPS **mutual authentication** schema between the device and the Snap4City framework is established. When a M2M connection from the IoT Device is performed, it is hard to retrieve the private keys of the IoT Devices, since they are stored in a key-secure storage (SIM card) commonly protected by an additional device password. In detail, the complete flow (see **Figure** 8) implies the creation of a private secret when the device is registered in the IoT Directory. Later, a digital certificate related to the private secret is digitally signed by the Snap4City framework. This signed certificate, with the private secret and the Snap4City certificate, are injected in the IoT Device in order to establish a mutual authenticated communication between the IoT Device and the platform, which is the IoT Firewall in this case.

In Snap4City, a **mixed authentication model with a moderated level of security** is available, when the IoT Device has very low resources and cannot support the complete SSL/TLS stack. It relies on the usage of the K1, K2 authentication system described above, plus the ability to verify by the IoT Device of the Snap4City endpoint by using the **thumbprint** (SHA, SHA3, Secure Hash Algorithms) of its public certificate to establish a secure HTTPS communication. This information can be recovered from the certificate metadata or can be easily estimated having the certificate only.

B. IoT APPLICATIONS VS SECURITY

As discussed in the abovementioned requirements and in the general architecture, IoT Applications exploit data access from storage, and connections with IoT Devices, Data Analytics, and general MicroServices. Thus, the Snap4City users can create IoT Applications with business logic using the Node-RED environment enriched by a large set of Snap4City nodes/MicroServices [65]. The IoT Applications can be executed on the cloud or on premise as IoT Edge Devices. Therefore, different security approaches have to be enforced to support all cases and scenarios.

When an IoT Application is executed on the cloud, it runs in a Docker container on an Apache Mesos & Marathon cluster. Web access to the Node-RED user interface is guaranteed via a reverse proxy configuration that follows the possible reconfigurations of the container cluster. An ad hoc authentication and authorization module has been developed to interoperate with the security management provided by Node-RED. A so-called *Strategy* has been written to plug the intelligence of the user access control to rely on the Snap4City infrastructure. Some adaptation of the Node-RED code was made to enable access to users with different kinds of roles. Any time a user accesses an IoT Application, the login is made using the same approach as any other Snap4City modules, retrieving from the SSO server a refresh

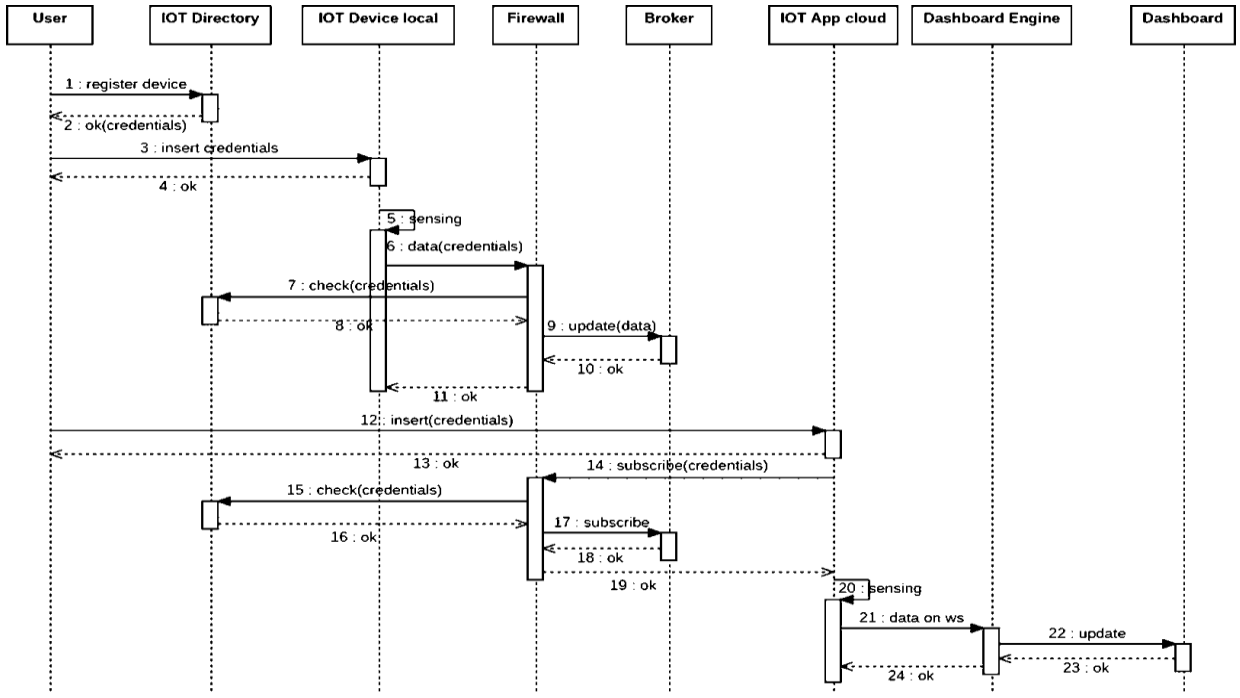

FIGURE 8. Sequence diagram of X509 certificate security enforcement.

token that is exchanged with an access token any time the IoT Application needs to communicate with any other Snap4City module. To avoid the user capability to login at any time, and thus, to cope with M2M communication, the refresh token elapses, and it is refreshed automatically by the platform in any 8-hour period. Since the IoT Application runs on a cluster and over time it can migrate between different nodes, the refresh token is stored in a local secure storage with an exclusive access of the specific IoT Application. Despite the flexibility, for security reasons, the users cannot load additional nodes autonomously from the Node-RED Library; they have to ask at the administrator to validate and load them. The cluster of Docker containers is also continuously monitored by the AMMA (Application and MicroService Monitor and Analyzer) tool for assessing the volume and messages exchanged on it [42].

In case the IoT Application is executed on premise/on IoT Edge Device, the User Interface is usually available only in a direct cable connection by the IoT Edge owner. In this case, if the business logic (IoT Application) written by the user programmers needs to access some functionality of the Snap4City on the cloud (i.e., MicroServices), the credentials of the user can be inserted manually and directly into the Node-RED flow, and the platform will take care to exchange them in favor of a valid refresh token and follow the same scenario previously described.

C. SECURITY OF IoT NETWORK VS DASHBOARDS

In the context of IoT, the communication modalities with Dashboards may be very complex since they are usually capable of recovering and sending data via multiple channels and sources. For example, they establish direct connections with: storage, IoT Brokers/Devices, IoT Applications (on the cloud and/or on IoT Edge), and some external service via Rest API, etc. These M2M connections present different modalities to establish authenticated and secure communications. In the context of IoT, the most interesting are those with IoT Devices/Brokers and IoT Applications since the others can be established as already described in the previous sections. Please note that both communications, namely, (i) IoT Devices/Brokers – Dashboards and (ii) IoT Applications – Dashboards, are all bidirectional, real-time, and data-driven modalities.

In Case (i), the secure or unsecure communication could be established using device/broker protocol and it may be difficult to acquire live updates on the user browser, for example, using MQTT, NGSI, etc., over TLS (the only solution would be to perform from Web client a polling on IoT Devices/Brokers). Thus, a solution that has been adopted has been to connect to an intermediate server that provides connection with Dashboards on the user browser via WebSocket secure, (WSs), in push. This solution has also been adopted for Case (ii), where IoT Applications/Node-RED presents Dashboard node MicroServices of Snap4City. WS protocol that allows data driven communications among the Internet browsers with the intermediation of a WebSocket Server. The secure communication is achieved by using a cluster of web socket-based applications that forward the messages to the proper connection. For example, a message produced by an IoT Application for a Dashboard's widget is taken and then forwarded to all the browsers that are currently viewing that

particular widget (a sort of WS Broker). Similarly, an input widget on a Dashboard produces a message that is forwarded to the IoT Application managing the widget. A JSON-based protocol has been developed to achieve the bidirectional communication with access control, where the access tokens are used to guarantee user identity and to check if the user is able to perform the requested action (unless it is a public dashboard).

VI. GDPR AND DELEGATION SYSTEM

At the moment of the registration, the user has to be informed about which Data Type (personal data of the person) will be collected. In addition to this notification, a signed consent is requested. All the collected Data Types need to provide a user interface for review, download and delete. The user may decide to delete them, while, for the law enforcement agency, the data would not be immediately deleted; they are moved and definitively deleted only after 30 days. In addition, the Snap4City platform permits the users to delegate access to their own data to any other user or anyone belonging to a specific group/organization, as already specified above regarding IoT Devices and their sensor's values. The delegated users have read-only access to each specific delegated data. A delegation can be any time reviewed and revoked, and it is not passed in cases of data change of ownership or cloning. A delegation can be created on top of the IoT Directory (IoT Device and sensors/actuators), Dashboard Builder (view of personal data) and IoT Application (user generated data).

A **pseudonymization** system management and the encryption of recorder data are designed by default to prevent identification of any specific individuals from compromised data in case of breach events. The Snap4City solution relies on storage where the links between the user's personal information and its data are stored completely decoupled via the use of a user's identifier shared between the MyPersonalData storage and the LDAP/CRM/Keycloak Snap4City's modules (*Pseudonymization*). In some cases, due to the large amount of data that can be generated, different storage systems could be involved to record the different user's data. The data storages are located in a set of proper servers, not directly accessible from any external access except from a set of authenticated MyPersonalData APIs. The Database is protected from external access via *Tablespace Encryption,* where any tables included in the MyPersonalData Database are protected by a set of keys recorded in the database itself. These keys are protected by an external Master key, memorized out of the database and accessible just by the system administrator (a superuser not registered in the Snap4City framework that is kept completely out the scope). Several techniques can be enabled on top of this separation (Master key rotation) to provide even more required security (*Encryption*). As already stated before, the in-transit encryption is guaranteed using always the SSL/TLS stack for network connections. Any modules of the Snap4City framework that need in some way to access some data for a specific use would need to use the authenticated APIs of the MyPersonalData module.

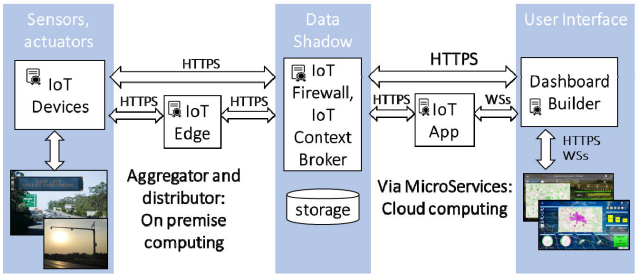

FIGURE 9. Chain of trust in Snap4City architecture.

A valid access token (in JWT format) has to be presented, that specifies the credentials of the user who requests the specific resource. The MyPersonalData eventually authorizes the access, if the module matches the ownership or the delegation of the specific requested data (*Access Control*). The detailed chain of trust of the Snap4City platform is highlighted in **Figure 9.**

In event of a data breach, the Snap4City framework is able, in a timely fashion, to detect and report on the issue and generate a set of records of what activities had been performed against the data. **Monitoring** in real time on different levels of detail is possible in the Snap4City framework by the system's administrator, via a system of notifications that mainly employ the sending of detailed emails. A set of managements tools are also able to provide constant monitoring of the different modules, services and databases' behaviors to proactively mitigate the risks. A set of thresholds and personal notification's messages are configured on the different analyzing's tools (*Monitoring and reporting*). Moreover, any Snap4City module is accompanied by an auditing user interface, where the activities performed on the modules and the requested service are graphically displayed for an easy and quick forensic analysis requested by the controller. Some other views on the activities on the modules are added for specific purposes, for example, a complete trace logging of the violation (request refused by the module) invoked on the MyPersonalData module (***Auditing***) (see **Figure** 10).

VII. EXPERIMENTAL RESULTS

The validation of the Snap4City platform has been performed by several teams testing and stressing the platform against more than 150 different test cases and scenarios that are listed on https://www.snap4city.org/108. The PCP took approximately 18 months for the final validation (third Phase of the PCP), including different kinds of validations for developers with Hackathon, with ICT officials for functional and non-functional requirements, for City Operators with dashboards and operative cases (see public Dashboards for Antwerp and Helsinki on Snap4City.org), for final users with mobile Apps and thousands of users, and included stress tests on the cloud and diffused Penetration Tests (PENTEST). Thus, Snap4City competed in the PCP with more than 25 different smart city platforms. Two mixed teams of the cities of

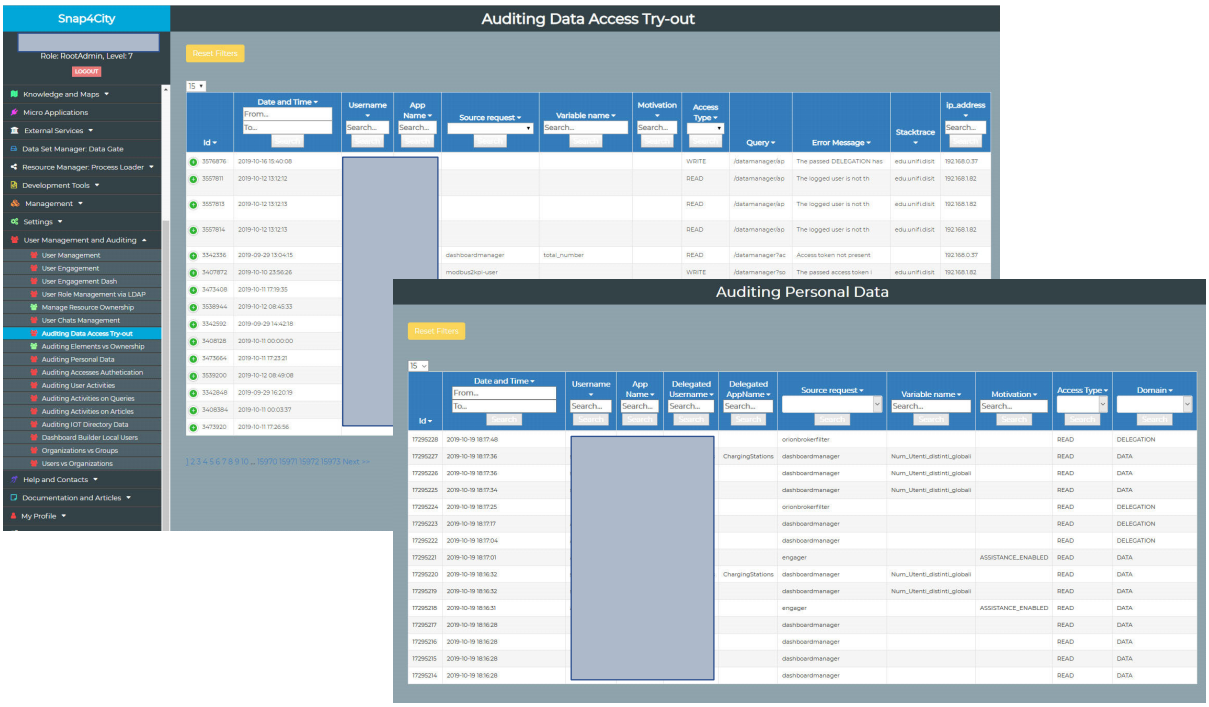

FIGURE 10. **Auditing violations on the access to MyPersonalData, and Try-Out. (some details have been obscured for privacy).**

Antwerp and Helsinki, with several people (including IoT experts, usability experts, ICT experts, and legal experts) have verified the aspects reported in the previous Table 2 point by point for the requirements, and in Table 3 for GDPR vs Requirements. Snap4City resulted in being the winning finalist of the many Smart City IoT Platforms, as described in web page: https://www.snap4city.org/558 and in which you can see the award, videos, and the link to the list of requirements that were overcome and validated, and in the web pages of Select4Cities. Moreover, the competition has been against requirements and their effective implementation and validation via test cases, all of which are accessible on www.snap4city.org.

A. GDPR COMPLIANCE VS REQUIREMENTS

The demonstration that a platform is GDPR-compliant is a very complex task, and there is lack of official tools for the verifications. On the web and market there are a number of checklists that can be adopted for the verification that a data management process is GDPR-compliant and only a few that may help the developers to test if their Web solution is GDPR-compliant. It should be noted that, some of the GDPR aspects may be evident from the user interface, UI, and thus they can be verified by external testers, while others (such as encryption, security level, etc.) can only be tested by code inspection and/or performing the penetration test (pentest) as described in Section VII-B. Interesting proposals have been suggested as a partial checklist on [60] and [61]. The list reported in **Table 3** reports the approach adopted for verification of GDPR compliance of Snap4City. In particular, it takes into account the UI aspects, source code access and demonstrations, as it was requested by Select4City to assess the platform.

Moreover, in Table 3, each major feature to be compliant with GDPR has been related to the proposed requirements of Section III. The detailed verification report would likely encompass some hundreds of pages and thus could not be accommodated to the space constraints of this article.

The set of GDPR features is not strongly related to the application domain of the smart city platform in terms of mobility, energy, home, etc., while it is mainly related to the usage of the platform, so that it refers to the applications. In all of the smart city domains mentioned, the user may be involved or not. An application on mobility for managing traffic without providing personalized services would not need to collect personal data; neither would it need to have citizens registered on the platform. The same can be stated for energy: the reporting of building energy consumption in an aggregated manner is not personal data. Therefore, the mapping of GDPR aspects vs domains would be very difficult to be realized without describing the applications. In Snap4City, the applications may all involve final users, in any domain. This implies that personal data have to be treated as described in the paper, disregarding the application domain. It is also very restrictive to think that the application would be domain-oriented for the data. For example, parking

TABLE 2. Requirements vs main sections of the paper and main modules of the architecture.

Requirement description	Security and Privacy Management IV.A IV.D	Ownership & Delegation IV.C VI	Personal Data IV.C VI.A	IoT Context brokers IV.C	IoT Directory IV.C	IoT Firewall IV.A	IoT Apps, and management V.B	Dashboard Bui8lder and management IV.C V.C	MicroServices IV.B	Smart City API IV.C	IoT Devices V.A V.C	IoT Edge, IoT App on premise IV.B V.B	Data Shadow and Storage IV.B IV.C	Knowledge Base Km4Clty IV.B	Snap4City Platform Support Living Lab IV.A IV.B
R1. IoT Brokers				X	X	X									
R2. IoT Discovery Abstraction					X					X				X	
R3. Authentication, Authorization	X	X	X												X
R4. Inform User of Security Level															X
R5. Developing Secure Applications	X	X				X	X		X						X
R6. Secure Communications	X	X													
R7. Open HW and Open SW									X		X	X		X	X
R8. Signed Content Vs Data Types	X	X													X
R9. Managing IoT Data Types	X	X	X	X	X	X	X	X	X	X	X	X	X		X
R10. Managing Ownership and Delegation	X	X	X												X
R11. Support Roles, Org. and Groups		X	X		X		X	X						X	X
R12. Encrypted Personal Data	X		X												X
R13. User Profile Management, forgotten			X												X
R14. Auditing Data Types	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
R15. Data Breach Detection	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
R16. Accounting	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
R17. Data Protection by Design	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
R18. Evidence of Level of Security	X	X													X

predictions are based on historical data of parking but also on traffic and environmental data, and events of people. This is the strong point of Big Data.

B. SECURITY AND PRIVACY ASSESSMENT

To assess the security and privacy of the solution, a number of penetration test's activities have been performed in the period April-July 2019 by professional companies of the sector. During the period, the Snap4City platform had approximately 1200 operative users on the Web Portal, 1.8 million of new data per day, 6 organizations that were operative (Florence, Helsinki, Antwerp, DISIT, Sardegna, and Garda), more than 4500 data ingestion processes executed per day, more than 320 IoT Applications running on the cloud and on IoT Edge Devices, more than 50 processes of Data Analytics, approximately 840 Dashboards of which 200 public, more than 300 MicroApplications of HTML5, 15 IoT Brokers, 5 Mobile Applications connected on Smarty City APIs, with approximately 3500 distinct active users on Mobile Apps daily, etc.

Any detected issues have been analyzed and for each of them a set of counter measurements have been taken to make the platform more robust to external intrusion and attacks. An approach of incremental tests has been chosen to spot the largest number of weaknesses and to enable a progressive evaluation-and-patching process. Each phase has been carried out by different actors such as:

- **first phase**: two groups of internal developers and security experts that know very well the Snap4City infrastructure;
- **second phase**: two groups of external testers that know nothing about the Snap4City infrastructure and that had very low interaction with the Snap4City developers;
- **third phase**: two third-party companies that executed the test without any interaction with the Snap4City developers; one of them has a high-ranking status and reputation on security issues and professionally works in the penetration test field of study.

During the period in which the penetration tests have been performed, an activity of a stress test was also carried out. Even if the focus of the pentest was not about analyzing how the platform responded to a high workload of requests, some feedback on the platform's performance has been collected to also improve this aspect of the platform and thus resilience on the workload and DoS (denial of service) attacks on APIs.

TABLE 3. Criteria for GDPR compliance verification features vs verification approach (verif.) and main requirements of section II (REQS.).

GDPR Compliance Verification Feature	Verif.	Reqs.
Signed consent	UI	R8
User profile management and control	UI	R13
Data Type private as default	UI	R8
Rights to access per element	UI	R9
Rights to transfer per element	UI	R10
Rights to erase per element and total	UI	R13
Rights to revoke/change per Data Type	UI	R10
An interface for Right management for Data Type	UI	R9
Clear Terms of Use and Privacy Policy	UI	--
Auditing Tools for Data Type	UI	R14
Publish as Anonymous	UI	R9
Encrypt personal users' data	Code	R12
Secure Authentication and Authorization	Code	R3
Data protection by Design	Code	R17
Secure connection	Code	R6
Security Control, data breach control, anonymization, etc.	PEN Test	R15, R16, R18

The penetration tests were performed following the following steps:

1. **Intelligence gathering activities against a target**: in this step, information about the target has been acquired using the OSINT (Open Source Intelligence) public accessible sources, and the information was collected using different tools such as: SpiderFoot (https://www.spiderfoot.net/), Maltego (https://www.maltego.com/), Shodan (https://www.shodan.io/).
- **Service detection and identification**: in this step, the target has been analyzed to find the services exposed and the versions of the used tools. This activity has been performed using tools such as: Necfraft (https://www.netcraft.com) on the main Snap4City domain, ZenMap to identify its open target ports (https://nmap.org/zenmap such as 80/tcp 443/tcp 5060/tcp 8080/tcp), Google Dork (https://securitytrails.com/blog/google-hacking-techniques), Nikto (https://cirt.net/Nikto2), DirBuster (https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project). The "theHarvester" script has been used (https://github.com/laramies/theHarvester) to retrieve any user's profiles leaked by the platform; no profiles were found, except the only one publicly presented as a contact point of the platform for external enquiries.
2. **Vulnerabilities detection, verification and analysis**: in this step, the target has been analyzed and deeply scanned for the different vulnerabilities using tools such as OWASP ZAP (https://www.owasp.org), Pentest-Tools and Burpsuite (https://portswigger.net/burp) and verified using sqlmap (http://www.sqlmap.org) and xenotix6 (https://www.owasp.org/index.php/OWASP_Xenotix_XSS_Exploit_Framework).

For example, the complete crawling of the identified contexts performed via the OWASP ZAP and w3af tools was performed from the point of view of (i) a user not registered and (ii) a logged user. Considering that any external links pointing to services out of the context have been excluded, a list of 73 ANAME records has been identified for a total number of approximately 8500 unique URLs that identified the attack surface where the following deeper analysis was performed.

A detailed set of attacks has been performed on domain and subdomains using the OSWAP ZAP and Arachni (https://www.arachni-scanner.com) tools. Even proceeding separately by subdomains, any complete analysis required several hours of computation. An analysis of the false positive alerts has been carried out to highlight just the true risks. For the main domain, 3095 URLs have been identified and a complete attack required more than 750 thousand requests for a total of approximately 39 hours of computation.

The pentests found some vulnerabilities that have been immediately solved, as described in the following and in particular:

- few "**SQL Injection**" vulnerabilities were found by using the OSWAP ZAP, Sqlmap (http://www.sqlmap.org) and Gobuster (https://github.com/OJ/gobuster) tools. These problems were found mainly in the *Dashboard Builder and management* modules. The problems identified were solved adding checks on the API parameters provided in the HTTP GET or POST requests; a specific list of 18 URLs (over a total of 244 URLs with high risk) has been identified and simulated attacks have been carried out manually and solved. The most relevant vulnerabilities were related to SQL injection of malicious code during the editing of information related to Dashboards and to Remote OS commands script injection permitted in a specific form of a User Interface exposed by the platform.
- a "**Broken authentication**" vulnerability was found as it was possible to do a password-guessing attack; to prevent this, it is set to not accept a login for 2 minutes after providing 5 wrong passwords.
- A few "**Sensitive Data Exposure**" regarding test users and passwords and a test private key present on GitHub were removed and changed; moreover, in some cases, the listing of directories was enabled;
- one "**Broken Access Control**" for a possible local file inclusion was found that was present in the CKAN tool used for managing open data. Moreover, in the process of fixing the SQL injection problems, also the potential broken access controls were identified that allowed a user to access data of another user by manipulating

the API parameters; in this case, stronger controls were added to prevent this situation;
- a few "**Security Misconfiguration**" vulnerabilities were found, and, in particular, were identified as a web server supporting the TLS1.0 and TLS1.1 protocols;
- some "**Cross-Site Scripting**" vulnerabilities were found, mainly in the *Dashboard Builder and management* module. The identified problems were solved escaping the input parameters using html entities when the input is not a html content; otherwise, only the *script* tag is sanitized;
- regarding "**Using components with known vulnerabilities**", the vulnerable versions of nginx and Apache2 web servers were detected and upgraded; moreover, a vulnerable version of Drupal 7 was found, which was at the basis of the "Snap4City Platform Support Living Lab" module, and it was upgraded to the latest version available.

For the other types of vulnerabilities, as mentioned in Section II.A, such as "XML External Entities (XXE)", "Insecure Deserialization", and "Insufficient Logging & Monitoring", nothing was found.

VIII. CONCLUSION

The shift of paradigm from completely central elaboration in the scenario of Big Data and the Smart City IoT toward a more distributed computation with edge computing has required different approaches in designing a platform that supports security by design. The Snap4City Smart City IoT architecture is composed of several modules that elaborate MicroServices that can be accessible for IoT Applications on the cloud and on premise, which is on the IoT edge. In those cases, security has to be addressed by design and by default about the privacy and protection of the managed data since in most cases, data can be very sensitive, due to their nature of city or personal data of the Living Lab users. Different levels of protections have to be offered on the basis of the sensitivity of the carried data and different requirements have to be addressed and satisfied in different use case scenarios (e.g., medical assistance, engineering and architecture, entertainment, city risk assessment, and city resilience). In this paper, Snap4City architecture and security solutions respecting the GDPR of the European Commission have been presented. The solutions addressed the full stack from IoT Devices, IoT Edge Devices on premise, IoT Applications on the cloud and on premise, Data Analytics, and Dashboards. Snap4City has been produced in response to the challenge launched by Select4Cities H2020 of the European Commission. Select4Cities identified a very large number of requirements for modern Smart Cities supporting IoT/IoE (Internet of Things/Everything) in hands of public administrations and Living Labs. In that challenge, Snap4City demonstrated to have satisfied all the requirements proposed. Moreover, innovative solutions have been proposed, and new specific requirements that were not identified since the beginning of the project have been outlined (and are reported in this paper). Therefore, as claimed by the evaluator, the Snap4City solution has gone beyond the expectations, namely, in the satisfaction of requirements including GDPR, on the innovations and with respect to the state of the art during the validation performed in the PCP with specific pilots and stress tests in Antwerp and Helsinki in Europe (they are top-level smart cities). The stress security assessment has been performed in a piloting period with more than 1200 registered users, thousands of processes per day, users on mobile apps, and more than 1.8 million of complex data ingested per day. The Snap4City architecture and solution described in this paper comply with the high security level and satisfy the GDPR of the EU. In the validation, Snap4City has also been stressed and tested by using several Penetration tests that allowed identifying a few vulnerabilities that have been solved in the current validated version. Thus, the solution guarantees an IoT end-2-end high level of trust for the current supported technologies in terms of security and privacy aspects, addressing security in the stack that includes: IoT Devices, IoT Edge Devices, IoT Applications (on the cloud and on IoT Edges), Data Analytics, dashboards, and Smart City APIs for Mobile Apps in which the security level is also supported. Snap4City is 100% open source and license/patent free, and it also currently in use in several cities in Tuscany (central Italy including Florence), in Antwerp, Helsinki and Lonato del Garda. Snap4City is a solution produced in response to a research challenge launched by the Select4Cities H2020 research and development project of the European Commission. Select4Cities identified a large number of requirements for modern Smart Cities supporting IoT/IoE (Internet of Things/Everything) in the hands of public administrations and Living Labs and selected a number of solutions. Therefore, at the end of the process that took 3 years of work, Snap4City has been identified as the winning solution.

ACKNOWLEDGMENT

The authors would like to thank the European Union's Horizon 2020 research and innovation program for funding the "Select4Cities" PCP project (within which the Snap4City framework has been supported), and all the companies and partners involved. Snap4City and Km4City are 100% open source technologies and the platform of DISIT Lab can be accessed at https://www.snap4city.org.

REFERENCES

[1] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT systems: Design challenges and opportunities," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2014, pp. 417–423.

[2] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, Aug. 2012.

[3] S. Li, L. Da Xu, and S. Zhao, "5G Internet of Things: A survey," *J. Ind. Inf. Integr.*, vol. 10, pp. 1–9, Jun. 2018.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st MCC workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[5] M. El-hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of Internet of Things (IoT) authentication schemes," *Sensors*, vol. 19, no. 5, p. 1141, Mar. 2019.
[6] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.
[7] P. Bellini *et al.*, "Smart city control room dashboards: Big data infrastructure, from data to decision support," *J. Vis. Lang. Comput.*, vol. 4, Dec. 2018, doi: 10.18293/VLSS2018-030.
[8] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Secur. Appl.*, vol. 38, pp. 8–27, Feb. 2018.
[9] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A standardization perspective," *IEEE Internet Things J.*, vol. 1, no. 3, pp. 265–275, Jun. 2014.
[10] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of Internet of Things IoT," *Int. J. Comput. Appl.*, vol. 7, p. 111, Feb. 2015.
[11] C. M. Medaglia and A. Serbanati, "An overview of privacy and security issues in the Internet of Things," in *The Internet Things*. New York, NY, USA: Springer, 2010, pp. 389–395.
[12] J. Voas, R. Kuhn, C. Kolias, A. Stavrou, and G. Kambourakis, "Cybertrust in the IoT Age," *Computer*, vol. 51, no. 7, pp. 12–15, Jul. 2018.
[13] K. Zhao and L. Ge, "A survey on the Internet of Things security," in *Proc. 9th Int. Conf. Comput. Intell. Secur.*, Dec. 2013, pp. 663–667.
[14] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Comput. Commun.*, vols. 89–90, pp. 5–16, Sep. 2016.
[15] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.
[16] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.
[17] C. Badii, P. Bellini, D. Cenni, A. Difino, P. Nesi, and M. Paolucci, "Analysis and assessment of a knowledge based smart city architecture providing service APIs," *Elsevier*, vol. 75, pp. 14–29, Oct. 2017, doi: 10.1016/j.future.2017.05.001.
[18] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Comput. Netw.*, vol. 141, pp. 199–221, Aug. 2018.
[19] D. Evans and D. M. Eyers, "Efficient data tagging for managing privacy in the Internet of Things," in *Proc. IEEE Int. Conf. Green Comput. Commun.*, Nov. 2012, pp. 244–248.
[20] T. Hammond, T. Hannay, B. Lund, and J. Scott, "Social bookmarking tools," *D-Lib Mag.*, vol. 11, no. 4, pp. 1082–9873, Apr. 2005.
[21] O. Tene, "Privacy Law's midlife crisis: A critical assessment of the second wave of global privacy laws," *Ohio St. LJ*, vol. 74, p. 1217, Feb. 2013.
[22] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, Feb. 1989.
[23] A. Alcaide, E. Palomar, J. Montero-Castillo, and A. Ribagorda, "Anonymous authentication for privacy-preserving IoT target-driven applications," *Comput. Secur.*, vol. 37, pp. 111–123, Sep. 2013.
[24] I. Chatzigiannakis, A. Vitaletti, and A. Pyrgelis, "A privacy-preserving smart parking system using an IoT elliptic curve based security platform," *Comput. Commun.*, vols. 89–90, pp. 165–177, Sep. 2016.
[25] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the Internet of Things," *Comput. Elect. Eng.*, vol. 37, no. 2, pp. 147–159, 2011.
[26] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2724–2737, Nov. 2013.
[27] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," SRI Comput. Sci. Lab., Tech. Rep. SRI-CSL-98-04, 1998, p. 19.
[28] A. Riahi, Y. Challal, E. Natalizio, Z. Chtourou, and A. Bouabdallah, "A systemic approach for IoT security," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, May 2013, pp. 351–355.
[29] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[30] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for IoT," in *Proc. 2nd Int. Conf. Internet-of-Things Design Implement (IoTDI)*, 2017, pp. 173–178.
[31] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.
[32] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. 1st Int. Conf. Mobile Syst., Appl. Services*, 2003, pp. 31–42.
[33] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Proc. Int. Conf. Pervasive Comput.* Berlin, Germany: Springer, May 2005, pp. 152–170.
[34] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, "Quantum privacy amplification and the security of quantum cryptography over noisy channels," *Phys. Rev. Lett.*, vol. 77, no. 13, pp. 2818–2821, Jul. 2002.
[35] S. M. Riazul Islam, D. Kwak, M. Humaun Kabir, M. Hossain, and K.-S. Kwak, "The Internet of Things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.
[36] T. Neubauer and J. Heurix, "A methodology for the pseudonymization of medical data," *Int. J. Med. Informat.*, vol. 80, no. 3, pp. 190–204, Mar. 2011.
[37] B. Claerhout and G. J. E. DeMoor, "Privacy protection for clinical and genomic data: The use of privacy-enhancing techniques in medicine," *Int. J. Med. Inform.*, vol. 74, nos. 2–4, pp. 257–265, Mar. 2005.
[38] I.-R. Chen, J. Guo, and F. Bao, "Trust management for SOA-based IoT and its application to service composition," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 482–495, May 2016.
[39] Y. Ben Saied, A. Olivereau, D. Zeghlache, and M. Laurent, "Trust management system design for the Internet of Things: A context-aware and multi-service approach," *Comput. Secur.*, vol. 39, pp. 351–365, Nov. 2013.
[40] P. Ray, "A survey on Internet of Things architectures," *EAI Endorsed Trans. Internet Things*, vol. 2, no. 5, Dec. 2016, Art. no. 151714.
[41] P. Nesi, C. Badii, P. Bellini, D. Cenni, G. Martelli, and M. Paolucci, "Km4City smart city API: An integrated support for mobility services," in *Proc. IEEE Int. Conf. Smart Comput.*, May 2016, pp. 1–8.
[42] P. Bellini, F. Bugli, P. Nesi, G. Pantaleo, M. Paolucci, and I. Zaza, "Data flow management and visual analytic for big data smart city/IOT," presented at the 19th IEEE Int. Conf. Scalable Comput. Commun., SCALCOM, Leicester, U.K., 2019.
[43] AWS. *Amazon IoT*. Accessed: Jan. 20, 2020. [Online]. Available: https://aws.amazon.com/iot/
[44] *MS Azure IoT*. Accessed: Jan. 20, 2020. [Online]. Available: https://azure.microsoft.com/en-us/free/iot/
[45] *Google IoT*. Accessed: Jan. 20, 2020. [Online]. Available: https://cloud.google.com/solutions/iot/
[46] *Node-RED*. Accessed: Jan. 20, 2020. [Online]. Available: https://nodered.org/
[47] *Fi-Ware*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.fiware.org/
[48] D. Balsamo, A. Elboreini, B. M. Al-Hashimi, and G. V. Merrett, "Exploring ARM mbed support for transient computing in energy harvesting IoT systems," in *Proc. 7th IEEE Int. Workshop Adv. Sensors Inter. (IWASI)*, Jun. 2017, pp. 115–120.
[49] P. Persson and O. Angelsmark, "Calvin–merging cloud and IoT," *Procedia Comput. Sci.*, vol. 52, pp. 210–217, Jun. 2015.
[50] *Home Kit Apple*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.apple.com/ios/home/
[51] *AirVantage IOT Solution*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.sierrawireless.com/products-and-solutions/sims-connectivity-and-cloud-services/iot-cloud-platform/
[52] *SmartThing Samsung*. Accessed: Jan. 20, 2020 [Online]. Available: https://smartthings.developer.samsung.com/
[53] *Kaa*. accessed: Jan. 20, 2020. [Online]. Available: https://www.kaaproject.org/
[54] *Carriots IOT*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.carriots.com
[55] *CISCO IOT*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html
[56] *IOT Eclipse*. Accessed: Jan. 20, 2020. [Online]. Available: https://iot.eclipse.org/
[57] *IOT Ignite*. Accessed: Jan. 20, 2020. [Online]. Available: https://devzone.iot-ignite.com/documents/

[58] *Bosch*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.bosch-iot-suite.com
[59] *ThingWorkx, PCT*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.ptc.com/it/products/iiot/thingworx-platform
[60] *Siemens MidSphere*. Accessed: Jan. 20, 2020. [Online]. Available: https://new.siemens.com/global/en/products/software/mindsphere.html
[61] *GDPR Compliance for Apps*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.privacypolicies.com/blog/gdpr-compliance-apps/
[62] *15 Key Steps to Make Your Software GDPR Compliant*. Accessed: Jan. 20, 2020. [Online]. Available: https://www.mindk.com/blog/how-to-make-your-software-gdpr-compliant/
[63] T. Pereira, L. Barreto, and A. Amaral, "Network and information security challenges within Industry 4.0 paradigm," *Procedia Manuf.*, vol. 13, pp. 1253–1260, Jan. 2017.
[64] D. Preuveneers, W. Joosen, and E. Ilie-Zudor, "Data protection compliance regulations and implications for smart factories of the future," in *Proc. 12th Int. Conf. Intell. Environ. (IE)*, Sep. 2016, pp. 40–47.
[65] C. Badii, P. Bellini, A. Difino, P. Nesi, G. Pantaleo, and M. Paolucci, "Microservices suite for smart city applications," *Sensors*, vol. 19, no. 21, p. 4798, Nov. 2019, doi: 10.3390/s19214798.
[66] *GDPR*. Accessed: Jan. 20, 2020. [Online]. Available: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

CLAUDIO BADII graduated from the Department of Computer Engineering, University of Florence, where he is currently pursuing the Ph.D. degree. He is currently a Research Fellow with the DISIT Lab (DINFO), University of Florence. He worked in the past to develop a system that simulates the behavior of a cloud platform. He currently focused on the development of mobile applications in the smart city and big data.

PIERFRANCESCO BELLINI received the degree in computer engineering and the Ph.D. degree in computer engineering and telecommunications from the University of Florence. He is currently a Professor and a Researcher with the Department of Information Engineering, University of Florence, and also a Professor of operating systems. His main research interests include semantic computing, ontology engineering, and cloud computing.

ANGELO DIFINO graduated from the Department of Computer Science, University of Turin, in 2000. He is currently pursuing the Ph.D. degree with the DISIT Laboratory, University of Florence, working on machine learning technologies capable of achieving scenarios of assistance/engagement. He has worked in the past with personalization and multimedia technologies in the MPEG.

PAOLO NESI (Member, IEEE) is currently a Full Professor with the Department of Information Engineering, University of Florence, and the Chief of the DISIT Lab and the Research Group. His research interests include massive parallel and distributed systems, big data, semantic computing, the IoT/IOE, smart cities, security and safety, and data analytics. He is a member of ACM, IAPR, AIIA, and CINI. He has been the General Chair of the IEEE SCI, the IEEE ICSM, the IEEE ICECCS, DMS, and international conferences, and the Program Chair of several others. He has been the Coordinator of several research and development multipartner international research and development projects of the European Commission, such as Snap4City H2020, RESOLUTE H2020, ECLAP FP7, AXMEDIS FP7, WEDELMUSIC, MOODS, and MUSICNETWORK. He has been involved in many other projects as a Coordinator or a Partner Coordinator, including Sii-Mobility, Km4City, REPLICATE H2020, TRAFAIR CEF, and WEEE Life. He has been a Co-Editor of MPEG SMR.