2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation

Privacy and security aspects on a Smart City IoT Platform

Claudio Badii, Pierfrancesco Bellini, Angelo Difino, Paolo Nesi
Department of Information Engineering, *University of Florence,* Florence, Italy
https://www.disit.org, https://www.snap4city.org , https://www.km4city.org
claudio.badii@unifi.it, pierfrancesco.bellini@unifi.it, angelo.difino@unifi.it, paolo.nesi@unifi.it

***Abstract*—Internet of Things paradigm enables computation and communication in tools that every day everyone uses. The vastness and heterogeneity of devices and the ways they are composed to offer innovative services and scenarios require a challenging vison in interoperability, security and in managing huge quantity of data. Many IoT frameworks and platforms propose to solve these issues, aggregating different sources of information and combine their flow of data in innovative services. Due to the potentially very sensible nature of some of this data, privacy and security aspects have to be taken into account by design and by default. An end-to-end secure solution has to permit the final users to have full control on their personal data and, on the other side, the framework has to support developers in writing applications offering the highest level of security/protection on their data flow. European Commission GDPR also added complexity to this context. In this paper, Snap4City solution to support such level of privacy and security in an IoT scenarios is presented. Snap4City has been developed in the context of Select4Cities PCP project of the European Commission.**

Keywords— IoT, smart city, security, end-2-end, GDPR.

I. Introduction

Internet of Things (IoT) could become a disruptive technology in the very close future, especially for citizens living in big metropolitan areas: the pervasiveness of the electronic IoT Devices, integrated in common "goods" people use every day (pens, tables, fridges or cars), is becoming deeper and deeper. In the future, the available addresses' space for these devices will be wide to be able to point to any sensors of any devices at any moment without (theoretically) no restrictions or constraints. Recent products that implements *Low-Power Wide Area Networks* (LPWAN) technologies for IoT introduced by Sigfox and Semtech (LoRa) have been gaining interest and have been under intense research and deployment campaigns worldwide [1]. At the same time, *short range* IoT commercial devices (based on technologies such as IEEE 802.15.4 or Bluetooth Low Energy [2]) are sold in increasing quantities and are already able to support scenario for smart homes, cities and industrial automation. Moreover, the start of the commercialization of *5G devices* and services are creating high expectation in networking new technologies, as killer application of previous technologies in metropolitan areas [3].

A wide variety of objects (e.g., smart bulbs, IP cameras, alarm clocks, buttons) are already today part of the user's environment, relaying on house's modem and hub devices to connect them over the network. The IoT infrastructures permit to realize 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) enable devices to connect each other and to exploit cloud-fog infrastructures [4]. The support, usually provided by IoT frameworks, entails to identify a structure which coordinates and controls processes conducted by several different IoT elements. These frameworks are a set of modules, rules, algorithms that organize the ways in which data are exchanged, processed and managed among all involved parties (devices, users, cloud, containers, edges). Even more, it supports the implementation of IoT applications that permits definition of user-component-logic, hiding at the same time the complexity of the infrastructure [5].

The architecture model of these IoT platforms must to be taken strongly in consideration, due to the huge amount of the data exchanged among parties and the complexity and the heterogeneity of the protocols and devices involved [6]. The information sent/received by the IoT devices could carry very sensible and private information; therefore, protection and cryptography techniques should be exploited and diffusely implemented. All steps of a secure communication must be guarantee as well as how the data are recorded and managed during all the cycle-life, from sensing, to injection, to consume for data analysis, to visualization [7]. In this context, the design of a flexible architecture is required to support the users to delegate the computation of personal data in a *safe environment* [8]. High level of *security* has to be assured in scenarios where the users completely rely on the system managing their data, that can be eventually exploited to help-inform-assist the data owner in daily activities and tasks, in a modern context with huge information overload [9].

The *complexity* of the IoT architectures makes hard to offer the needed level of security: the computation is becoming more and more *world wide distributed* among peers and not anymore centralized in a controlled static environment or cloud data center. The necessity to support *heterogeneous source* of information (IoT devices and sensors), accessed in a multitude of different ways using *various communication systems*, bringing another level of complexity, in scenario where it is difficult to predict when and where the data are generated and made available (ingested) to the system/subsystems [10]. Lacking regulations and certifications for devices and sensors as such as for the complete IoT infrastructure leaves to closed systems the ability to lobbing on proprietary solution.

In section II, the major requirements needed for a robust IoT platforms are listed. In section III, the Snap4City solution is presented, with some introduction on the architecture and focusing on aspects of users' data privacy and security. In section IV, conclusions to the problems highlighted and solved by Snap4City solution are draw.

978-1-7281-4034-6/19/$31.00 ©2019 IEEE
DOI 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00250

II. REQUIREMENTS

The list of requirements an IoT platform has to satisfy in term of privacy and security can be long and endless. The major issues identified in design of Snap4City solution are reported as follows.

- the platform has to manage the data as **sensible data** by default (both single elements and data sets, time series, etc.). Data can be explicitly linked to the users or to users' devices. It has to be possible to specify the time's interval for which the data have to be stored, or if that is not possible, the criteria used to determine this interval;
- the platform has to request **explicit consent** to any of the data/dataset identified above. "Informed consent by default" in no valid anymore in Europe according to GDPR. The platform has to provide clear information about each data/dataset license (e.g., how long and eventually how it can be shared to third parties, such as users, groups, organizations);
- the platform has to provide the user the "*right to access*" and "*right to be forgotten*". A user must be able to fully manage own data and eventually requiring the **erasure** of them. The platform has to permit access on data by law enforcement agencies/police, for a specific limited time interval;
- the platform has to provide the users the "right to review" (**auditing**). It should be possible for the users to **check** the consents provided and about how automated decisions are computed on personal data collected. The user has to be able to review the complete log of the accesses made on private data (when, why, who request data);
- the platform has to implement methods of **data breach detection** to disclose in a short period whenever some data has been tampered or leaked;
- the platform has to support **data protection** (privacy and security) by design and by default, requiring controls built into products and services from the earliest stage;
- the platform has to provide a way to define **roles and responsibilities** within organizations to permit different level of user's credentials; the platform has to implement appropriate technical and organization measures to ensure a **level of security** appropriate to the risk, including, and not limited to pseudo-anonymization, confidentiality and integrity, availability, resilience, disaster recovery, periodic stress testing and workload;
- the platform has to provide a system for **unique entity management, authentication and authorization**. Just the user that is authorized can access its private data and eventually, if a user has provided a consent to another user/process/entity, it should be possible to enforce this access;
- the platform has to store the data in an **encrypted** way, to prevent in the event of breach, the identification of any specific individual compromised data;
- the platform should support **accounting** in terms of reporting resource consumption and billing, if any;
- the platform has to assure a tolerable security level even for device with **constrained resources** (energy consumption and memory availability) and belonging to different hardware specification (e.g., microcontroller, microprocessors, personal computer, mobile devices). The platform has to provide a different level of security based on the different level of sensitivity of the carried data;
- the platform has to **avoid** exposing a **single point of failure**;
- the platform must support protection of data in any possible **different configuration** it can be deployed: local computation, on-cloud scenario, mixed between local and cloud (the stack IoT Device, IoT edge, IoT app on cloud, dashboards, etc.), sensor's data retrieved in push, pull, real-time, etc.

Beside the security aspects, it is important to keep in mind more general requirements valid for any IoT architecture:

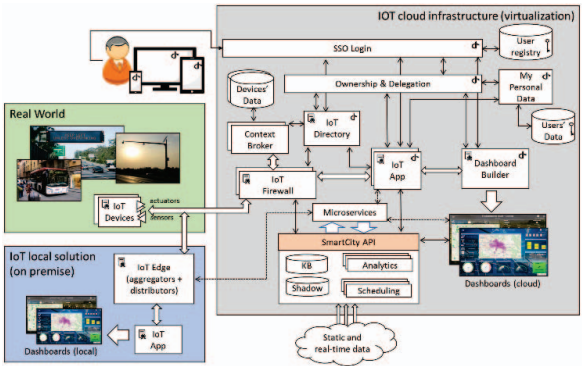
- Cloud-Fog data routing and local (on premise) IoT computation on IoT edge;
- Abstraction of IoT brokers and devices with support provided by IoT Directory and Knowledge Base;
- Device Shadowing and management (IoT data storage); heterogeneous in/out sensors-actuators;
- Programming and data analytics abstraction, exploiting IoT data and historical contextual data;
- Scalability and cloud service for Context Brokers, IoT devices, Container, IoT Applications;
- Dashboards and data presentation, business intelligence, visual analytics (Web-Socket support);

III. ARCHITECTURE

The Snap4City architecture has been designed and implemented using principles depicted above by the Cloud-Fog IoT architectures, decoupling IoT edge functionalities (functionality directly accessed on the user premise) from the core platform services. The architecture is flexible enough to support the above requirements which are valid for a wide range of IoT use cases with for local computation, cloud computation and mixed. In Snap4City, the set of tools are able to guarantee the privacy and protection of the data managed by the system are enforced in main architecture (in respect mainly to the GDPR, the ENOLL and the Select4Cities set of recommendations).

Fig.1. Snap4City main architecture

From the upper left part of Figure 1, the system permits



the user to access the platform with a set of devices via a User Interface and via a set of conformant mobile applications. The user accesses the platform functionalities in an authenticated

way using its credentials. His identity, whenever verified, is propagated by the **SSO Login** module towards any Snap4City module that need to access some user's data. The users' characteristics (mandatory and optional information about the user) are kept in a separate storage called User Registry. The platform is interfaced to the **real world** via a set of IoT Devices, sensors and actuators. A multitude of different devices are supported: simple **IoT devices** based on *microchips* (e.g., the esp8266 with microcontroller capabilities); more powerful devices based on *small single board computers* (e.g., Raspberry PI) with better capabilities that can be configured with sensors/actuators onboard to act as aggregator of other IoT devices on their premise (playing the role of an IoT Edge) or as router towards the Snap4City backbone (IoT Gateway); *handheld* personal computer (e.g., Android on smartphones or similar devices) with embedded strong network capabilities; any *personal computer* and *Internet devices*; preconfigured *virtual* device created directly on the cloud infrastructure.

Any of the supported IoT devices can communicate directly to the platform whenever they have an Internet communication. Otherwise, they can eventually be configured to be connected with the Snap4City backbone using an aggregator/distributor device (**IoT Edge**) connected for example via a wireless network interface towards an Internet access point using standard communication capabilities (IEEE 802.x) or other private/patented technologies (LoRa, SigFox, RS485, ModBUS, BLE).

The IoT Edge can be also configured to dispatch the message locally towards a set of IoT Applications installed directly on the premise of the user's IoT Devices. The idea is to enable Fog computation completely decoupled from the Snap4City backbone. With this kind of configuration, the data can be elaborated and presented locally (in terms of closeness to the installed IoT Device and IoT Edge) via a set of Dashboards, created by the user via platform's tools, and instantiated locally on devices on their premises. In some case these IoT Applications can make use of External Services provided by a set of **MicroServices** exposed by the platform: they are accessed remotely via Internet connection and clearly in case the connectivity is not available or guarantee, the application logic has to treat this kind of information according to a backup ad-hoc solution. The MicroServices are implemented using the **Smart City API** [11] that work directly on top of a Km4City based **Knowledge Base** (KB) and a Data **Shadowing system**. A set of additional tools are also available; for simplicity here just two main macro-blocks are presented: the **analytics** set of tools and the **scheduling** functionality. The KB of the microservices is kept updated in real-time via the injection of **static and real time data** retrieved by a set of ETL processes, scheduled in a proper way basing on the nature and the dynamicity of the same data.

In most of the IoT cloud solutions, the data flows to/from a set of IOT **Brokers** (Mosquito MQTT, Orion NSGI, etc.) to enable a shadowing system. In front of the Context Brokers an **IoT Firewall** enables the access just to granted device/entities. The data collected are stored in repositories and made available to the other internal components of the Snap4City solution, mainly to the IoT Applications deployed on cloud. The IoT Applications (based on Node-RED plus Snap4City library of nodes for Smart City) permit the users to design in an easy graphical way an ad-hoc application logic that manages data and provides applicative users interface via the **Dashboard** components. These Dashboard nodes of IOT Applications allow at the final user to compose in an intuitive way IoT App visual interface (input and output) by using a list of widgets that can be defined on top of raw or structured data, for full user interaction including virtual sensors and actuators.

The IoT Context Brokers are registered and managed by an **IoT Directory:** a module to manage the IoT Device's registry and associate the user credentials to IoT Device, so as the access can be granted. Different levels/kinds of credentials are available (key credentials, certificate credentials) and are specified whenever a new device is registered by the user in the Snap4City IOT Directory. The data carried by the IoT Device can also flow directly to IoT Applications. Any IoT Device, sensor and actuator, Dashboard and IoT Application whenever instantiated/created are associated with an owner. Therefore, any Snap4City module can enable/deny access to specific user/group/organization, which can be the recipient of a delegation/right. The **Ownership** module has to be acceded in an authenticated way to provide access information. Beside the concept of ownership, the mechanism of **Delegation** (consent) is also implemented to permit users to delegate other users, or groups/organizations, to access to their private entities (data, values from sensors, dashboards). A user can also make public some of entities, creating a special "*Delegation to Anyone*" in an anonymous form, to give access to any users to data/resources, without revealing himself. The information of the ownership and the delegations, beside any other personal data a user like to keep in a private and secure storage are managed by the **MyPersonalData** module, where data are managed/stored in a protected way, in conformance to the GDPR. Any personal's data is labelled using a general classification (motivation, variable name, variable value, variable unit). Therefore, developers can create their own data sets and eventually display them with Dashboard Widgets. Beside the personal data, the **Key Performance Indicator manager** (KPI) is also available, in order to manage ad-hoc time series and enrich them with more structured metadata (GPS location, healthiness, …).

The way sensors and actuators are connected to the Snap4City backbone and which IoT Infrastructure's services are involved strongly depends on the use case scenario to implement, on the energy and connectivity available around the sensors and on the level of security required (sensibility of the transmitted data). The Snap4City IoT cloud infrastructure is the backbone of the solution. It is completely virtualized and acts as the boilerplate where City Operators configure, via a set of graphical tools, processes for data aggregation and data analytics. The data generated by IoT Applications can be also injected back in the internal Knowledge Base and exposed to 3rd party modules/dashboards for graphical presentation or for further analysis, keeping in mind that just the data owner or delegated users will be enabled to access them.

A. User Access Security

The Snap4City solution uses several tools for **authentication and authorization** enforcement and users/entities access to platform resources (see Figure 2). The users' registry is partially managed by a **distributed directory information service** (LDAP) and partially maintained in a **customer-relationship management** (CRM, Drupal). Mandatory information is saved in LDAP and is composed by usernames, hashed version of the users' password, emails, roles and organizations/groups affiliations, ... The username is the primary key to identify the users in the

Snap4City platform; the role (Manager, AreaManager, ToolAdmin and RootAdmin) is used to classify the users regarding their level of trust and to enable different level of details to access the Snap4City functionalities (e.g., enable different views of the interface, permits more or less data investigation). Moreover, some tools may be accessed by not registered users. The organization/group information is used to organize the users in term of location or purpose (for organization, e.g., Firenze, Helsinki, Antwerp) and affiliation (for group, e.g., Developer, Services, General Management). The LDAP module is disconnected from the Internet and is reachable just from an internal subnetwork proper configurated.

Optional information regarding the user is saved in the CRM and includes names and surnames, registration and last access dates, locations in term of residency's country and its time-zones. Any of this information is pseudo anonymized with technique as specified below.

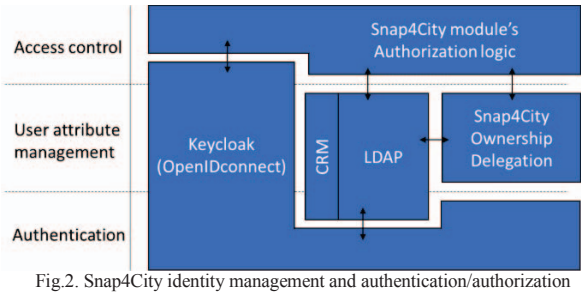


Fig.2. Snap4City identity management and authentication/authorization

Via the CRM Web Interface, the user can register into Snap4City ecosystem specifying a username, a valid e-mail address and his password. The users can anytime review and update his information and eventually remove completely his account. A CRM plugin permits to keep synchronized the user's information with the LDAP module. The CRM is installed on a Debian Virtual Machine accessible from the Internet through an TLS/SSL entry point with a proper signed certificate and is kept updated and live monitored.

The user authentication is enforced using a **Single Sign-on (SSo)** system and an **Identity Management** functionality (Keycloak). The users' registry is kept synchronized with a LDAP module and uses the OpenIDConnect protocol to provide a users' authentication system to any Snap4City modules (which include also the CRM module). Anytime a user tries to access at a resource exposed by a Snap4City module, when the user is not already logged, the request is forwarded to the SSO Server that requires to the user to specify his credentials. Whenever there is a match to the credentials stored in the LDAP users' register, the user is *eventually* granted to access the resource exposed by the Snap4City module (depending on ad-hoc additional enforcement provided by the modules): the user's role specified in LDAP is mapped in the SSO rules' registry, so the system administrator can specify in detail particular access rule to enable or deny access (specification of grants) to specific user's role to specific Snap4City modules. During the authentication phase of the user, Keycloak passes to the contacted Snap4City modules, beside their usernames, also their roles; thus, whenever the user is granted to access a specific module, the module itself can implement internally another finer ad-hoc authorization's rule to eventually permits/denies the requested resource/functionality. The module could also contact the LDAP to directly retrieve the organizations/groups the user belongs and the Ownership/Delegation Snap4City module to enable a transversal authorization schema.

Any communications between the 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 (in this solution a JWT -JSON WebToken- is used), permits a SSO system among the different modules. When the user accesses a resource from a module in an *interactive* manner via a Web interface, whenever a JWT is not present or not valid (elapsed, tempered or not correctly digital signed), the user is always redirected to the login Web page. The Keycloak is configured to provide authentication via the OpenIDConnect protocol, so the OAuth 2.0 protocols is involved for the authentication part enriched on top with the user's identification part.

In IOT connections and when Machine to Machine connections are needed, an entity (like an external machine/device) need to access a Snap4City module in a *programmatic* manner, a formal user is still needed to specify credentials as above. In those cases, an offline-access token is released (refresh token), and the machine does not need to request the user to login anymore and can use this refresh token to request a normal access token as specified above. This refresh offline token has a longer-live time so human interaction is limited. Once the authentication and identification of the user is successfully completed (and a valid access token is returned), the Snap4City modules are contacted attaching the JWT as their credentials. They will be used to validate if the user/machine has enough rights to access the requested data/resource.

The OpenIDConnect protocols enables the separation of entry points and communications for the different Snap4City modules to authenticate the users, even if the shared JWT can enable the access to different modules. This configuration permits a nice user experience, enabling at the same time the possibility of creating a detailed auditing on the user access modality and frequency. Moreover, the secret the different Snap4City modules need to specify to access the SSO server for user's authentication are specific to any module, thus, in case of leaking or intrusion by a malicious user, the problem can be isolated without the need of a complete system reconfiguration, and without exposing a single point of failure.

B. Network Access Protection

In order to enable access from/to IoT Devices (and its related sensors and actuators), a user has to register the device into the Snap4City ecosystem. Thus, any communication between the device and the platform can be made in a protected and authenticated manners. The registration is made using a Snap4City module called IoT Directory: it exposes a Web Interface to the user that has to be authenticated and authorized as described in the previous section. For registration, the user specifies several information regarding its device as unique identifier, type/manufactory, location to be chosen on a map and the endpoint broker (chosen by a set of possible registered configurations). In case the number of the devices to be register is massive, the user can use a functionality that automatically register (in a background modality) a large set of devices via a detailed csv file containing all the needed information (Bulk Registration). A device can be registered using a pre-compiled template (model of the device), by using automatically filled forms: the model and the device type also influence the level of the protection

the platform can guarantee for this particular hardware/device. The Snap4City framework supports different authentication schemas for different kind of IoT Devices and level of protection, including proprietary solution beside open-standard ones as described in the following.

For the **proprietary solution**, the data flow and the authentication systems are completely demanded to their proprietary infrastructure. In this case, the IoT Device sends the sensor's data directly to their proprietary Context Broker in their supported protected way (e.g., for Sigfox configuration, it relies on credentials K1, K2), meanwhile the Snap4City platform is configured to retrieve the data (in pull-up modality, on demand) in a second moment, whenever the user specifies the needed credentials for the proprietary solution. Also push modality can be used from SigFOX. The data injected in the Snap4City solution is labelled with the name of the IoT Device owner that generate data and are managed in the same way as specified below.

For **custom devices,** the platform can rely on several communication protocols like NSGI, MQTT, COAP and AMQP and provide a Snap4City protection system. If NGSI is chosen, the IOT Orion Context Broker of Fi-Ware is used, with an addition of flexible security layer developed by the authors. Since the IOT Orion Broker does not support secure connections. In the proposed solution, the first basic authorization modality makes use of a **couple of keys** (K1, K2) as credentials, that are automatically generated by the Snap4City framework and associated with the device at the moment it is registered in the IoT Directory. These credentials are made available just to the owner of the IoT Device for future inspection. Whenever the IoT Device likes to communicate with the Snap4City framework (in case a value of a sensor is updated), it simply needs to enrich his communication channel with these K1, K2 (credentials) beside his identifier (the same apply to any IoT Application that like to send a new value to an actuators). These keys have to be specified by the user in the configuration of the IoT Device/IoT Application.

An additional set of keys are generated in case the owner of the device chose to delegate another user, or group of users/organizations/groups, to access the values provided by the owned IoT Device/Sensors (in read only modality). Whenever the delegation is removed, the K1 and K2 are erased and will not be any more valid. When an IoT Device has to be declared "Public", an *Anonymous* delegation is generated. In this later case, the K1, K2 can be and are omitted, and reading operation is always permitted. Only the owner is enabled to modify the internal status of the device/variables.

For a higher level of security, the IoT Device owner can choose a stronger (in term of security) authentication solution based on **X.509 certificate**. This technique involves the exchange of a signed digital certificate, based on a private-public key cryptography solution. The Snap4City IoT Directory acts as 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 direct access to the device is performed, it is hard to retrieve the private keys of the IoT Devices, since it is stored in a key-secure storage (SIM card) commonly protected by an additional device password. In detail, the complete flow [see Figure 3] 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 have to be injected in the IoT Device in order to establish a mutual authenticated communication between the IoT device and the platform.

A mixed authentication system is also available whenever the IoT Device has very low resources and cannot support the complete SSL/TLS stack. It relies on K1,K2 authentication system plus the ability to verify by the IoT Device the Snap4City endpoint using the **thumbprint** (SHA, SHA3) of its public certificate, to establish a secure HTTPS communication.

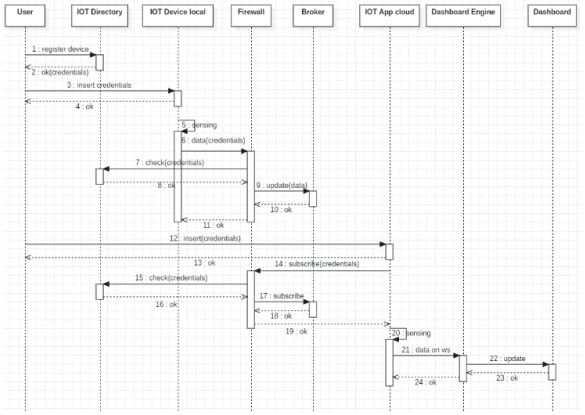


Fig. *3*. Sequence diagram of X509 certificate security enforcement

C. User Privacy: GDPR and delegation system

At the moment of the registration, the user is informed about which data are collected (mandatory information about the user specified in the CRM are needed to enable the service and cannot be avoided, due their importance to deliver the requested service to the user). Beside this notification, a signed consent is requested about the additional set of tracked data (***Identification of Impact to Personal Data***). Optional data information can be updated and reviewed via the CRM Web Interface. Any time the user can delete his account, taking into account that the user's data will not be made available to the user anymore (even if a time-frame is guaranteed for police's officer inspection).

Whenever the data from a user are injected in the system, they are recorded in a MyPersonalData Snap4City module, which is a secure storage completely decoupled by any other module. The user can review any time the recorded data using an authenticated Web Interface and eventually delete any recorded data sets or also forget completely any stored data. To enable inspection from law enforcement agency the data are not immediately deleted: they are marked to be deleted after a specific time-frame (usually 30 days). After this time-frame elapse, the data are completely erased from the system. The data marked to be delated are not available to the user, and just to the system administrator (***Retention of Personal Data***).

The stored data in the MyPersonalData module are by default set private of the IoT Device/Application owner that generated them. Therefore, the data owner is the only one that can access for inspection and analysis (beside users with the highest role of RootToolAdmin, trusted users' role granted just to system admin and law enforcement agency). The

platform permits the users to delegate in access any other user or anyone belonging to a specific group/organization. In this case, the delegated users have access to read specific delegated data. A delegation can be any time reviewed and revoked. A delegation can be created by using IoT Directory (IoT Device and sensors/actuators), Dashboard Builder (view of personal data) and IoT Application manager (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 a breach events. The Snap4City solution relies on a 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/Drupal/Keycloak modules (*Pseudonymization*). The data storages are located in a set of servers, not directly visible from any external access except from a set of authenticated MyPersonalData APIs. The Database is protected from external access via "*Tablespace Encryption*". Tables included in the MyPersonalData DB 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. Several techniques can be enabled on top of this separation (Master key rotation) to provide even more required security (*Encryption*). Any modules of the Snap4City framework, that need in some way to access some data for a specific user, needs to use the authenticated APIs of the MyPersonalData module that eventually authorizes the access, if it matches the ownership or the delegation of the specific requested data (*Access Control*). The detailed chain of trust of Snap4city platform is highlighted in Figure 4.

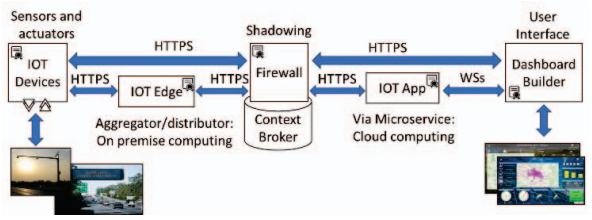


Fig. 4. Chain of trust in Snap4City architecture

In event of a data breach, the Snap4City framework is able, in timely fashion, to detect and report on the issue and also generate a set of records of activities 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 constant monitoring the different modules, services and databases' behaviors to proactively mitigate risks. A set of thresholds and personal notification's messages can be configured on the different analyzing's tools (*Monitoring and reporting*). Moreover, any Snap4City module is accompanied with an auditing UI, where the activities performed on the modules and the requested service are graphically displayed for an easy and quickly forensic analysis requested by the controller. Some other views on the activities on the modules are added for specific purpose, for example a complete trace logging of the violation (request refused by the module) invoked on the MyPersonalData module (***Auditing***).

IV. Conclusion

The shift of paradigm, from completely central elaboration in scenario of Big Data and Smart City towards a more distributed computation using a Fog-Cloud model requires different approaches in design platform and framework. These new Smart City IoT architectures, that can be composed by several modules and MicroServices, have to take care by design and by default about the privacy and protection of the managed data, since in most cases can be very sensitive, due to their nature of city data. Different levels of protections have to be provided on the basis of the sensitivity of the carried data and also different requirements may be defined for different use case scenarios (e.g., medical assistance, engineering and architecture, entertainment, city risk assessment, city resilience), and devices. In this paper, the Snap4City solution has been presented: its architecture has been described highlighting the components that enabled the platform to comply with high security standards and the GDPR of the European Commission. The Snap4City solution guarantees an end-2-end high level of trust for the current supported technologies in terms of security and privacy aspects, addressing security in the stack: IOT Devices, IOT Edge, IOT Applications, Data Analytics, and Dashboards.

Acknowledgment

The authors would like to thank the European Union's Horizon 2020 research and innovation program for funding the "Select For Cities" project (within which the Snap4City framework has been implemented) under grant agreement No 688196, and also all the companies and partners involved. Snap4City is an open technology and research of DISIT Lab.

References

[1] Xu, T., Wendt, J. B., & Potkonjak, M. (2014). Security of IoT systems: Design challenges and opportunities. In *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design* (pp. 417-423). IEEE Press.

[2] Gomez, C., Oller, J., & Paradells, J. (2012). Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *Sensors*, *12*(9), 11734-11753.

[3] Li, S., Da Xu, L., & Zhao, S. (2018). 5G internet of things: A survey. *Journal of Industrial Information Integration*, *10*, 1-9.

[4] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (pp. 13-16). ACM.

[5] Ammar, M., Russello, G., & Crispo, B. (2018). Internet of Things: A survey on the security of IoT frameworks. *Journal of Information Security and Applications*, *38*, 8-27.

[6] Keoh, S. L., Kumar, S. S., & Tschofenig, H. (2014). Securing the internet of things: A standardization perspective. *IEEE Internet of things Journal*, *1*(3), 265-275.

[7] Farooq, M. U., Waseem, M., Khairi, A., & Mazhar, S. (2015). A critical analysis on the security concerns of internet of things (IoT). *International Journal of Computer Applications*, *111*(7).

[8] Medaglia, C. M., & Serbanati, A. (2010). An overview of privacy and security issues in the internet of things. In *The internet of things* (pp. 389-395). Springer, New York, NY. Internet of Things - New security and privacy challenges.

[9] Voas, J., Kuhn, R., Kolias, C., Stavrou, A., & Kambourakis, G. (2018). Cybertrust in the IoT Age. *Computer*, *51*(7), 12-15.

[10] Zhao, K., & Ge, L. (2013). A survey on the internet of things security. In *2013 Ninth international conference on computational intelligence and security* (pp. 663-667). IEEE.

[11] Nesi, P., Badii, C., Bellini, P., Cenni, D., Martelli, G., & Paolucci, M. (2016). Km4City Smart City API: an integrated support for mobility services. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)* (pp. 1-8). IEEE.