Chapter 5
Designing Secure IoT Architectures for Smart City Applications

Elias Tragos, Alexandros Fragkiadakis, Vangelis Angelakis and Henrich C. Pöhls

5.1 Introduction

The Internet of Things (IoT) has received significant attention lately due to the numerous potential improvements it can bring to the everyday lives of peoples, simplifying many of their daily life activities. This is mainly evident in the domains of smart buildings, smart health, and smart cities. Especially in the smart city domain, the benefits for the citizens, the society, and the economy at large scale, have transformed the IoT into a major trend, and many municipalities are trying to build their city development strategies around it. Of course, a very first point is to build an IoT infrastructure that can be quite costly. Then, the cities have to develop applications that will run on top of this infrastructure and will address the requirements and the needs of the citizens. The latter part is very important in order to motivate the citizens to "accept" and adopt the IoT technologies.

A major factor for the citizens to adopt and use the IoT technologies in this direction, is the trustworthiness and the reliability of IoT as a whole. It is reasonable to assume that citizens can be sceptical for this new technology, especially when thousands or millions of new small and "invisible" devices can be around them all the time, with the potential to monitor continuously their everyday activities. These devices are not only becoming smaller, but also more intelligent, autonomous and active and are seamlessly integrated in the smart city environments. There are

E. Tragos (✉) · A. Fragkiadakis
Institute of Computer Science, Foundation for Research and Technology—Hellas, (FORTH-ICS), Heraklion, Greece
e-mail: etragos@ics.forth.gr

V. Angelakis
Communications and Transport Systems, Department of Science and Technology, Linköping University, Campus Norrköping, Norrköping, Sweden

H.C. Pöhls
Department of Informatics and Mathematics, University of Passau, Passau, Germany

V. Angelakis et al. (eds.), *Designing, Developing, and Facilitating Smart Cities*,
DOI 10.1007/978-3-319-44924-1_5

various projections that many billions of devices will be connected to the IoT in the next few years [1, 2]. This increasing number of autonomous connected devices that are not only monitoring their environment, but can also act upon it can become a worrying factor for the citizens, in terms of stealing their private information, logging their activities, monitoring their presence, or even harming their life. In order to motivate the users to adopt the IoT and use the smart city applications in their everyday activities, they have to be provided with guarantees that the IoT systems: (i) will be reliable in terms of the information they provide, (ii) will exchange the information in a secure way and (iii) will safeguard their private information.

To address the above requirements, the whole IoT system has to be secure in a cross-layer manner, starting from the physical layer and covering all layers up to the applications. Since IoT systems are based on Wireless Sensor Networks, various sensor security mechanisms can be adopted also in the IoT [46]. Security and privacy-preserving functionalities have to be embedded in the system's operation from the design phase, because post-mortem corrections and improvements can only cover some holes, but will not provide full-scale security. Thus, there is lately a move toward adopting the concepts of "security and privacy by design" when someone wants to build a new IoT system. As described in [3], the concept of "security by design" describes the need for considering all possible security issues at the conception phase, and designing respective mechanisms to prevent and react to these issues.

Although for adopting the concept of "security by design" a security toolbox might be enough, for adopting the concept of "privacy by design" a more holistic approach is required, since a set of privacy enhancing technologies cannot fully protect the users' privacy [4]. This requires embedding privacy concepts in the core of many system functionalities, e.g., onboard the sensors in order to be able to not collect identifiable information, at the gateways to be able to hide identifiable information before forwarding encrypted data, at the middleware to disallow the linking of information between different services, etc.

Apart from security and privacy, the issue of Trust in IoT is also attracting a lot of research interest lately. Trustworthy internet is defined in [5] as the Internet system that is secure, reliable, and resilient to attacks and failures. This means that the notion of Trust in the IoT has to be evaluated using metrics of: (i) reliability of both systems and data, (ii) security of the infrastructure and the provided services and (iii) ability of the system to prevent and respond to attacks and failures. An overall trust model for the IoT, since it incorporates security, and privacy, can be potentially very useful for promoting the IoT systems to end users or citizens. It is reasonable to assume that a system that is certified as "trustworthy" will be much more attractive for a user than any other system. However, in order to make a system trustworthy, its functional architecture has to incorporate all security, privacy, and trust functionalities.

In this chapter, we provide an overview of the requirements and challenges for designing secure IoT architectures. We also present previous attempts of many IoT-related projects to design and develop mechanisms and concepts for improving

the security, the privacy, and the trustworthiness of IoT. The focus is on the latest IoT projects and we try to provide a brief description of the main security and privacy features that they have embedded on their architectures. The chapter will conclude with a brief discussion on the common functionalities of the projects and the open research items providing some suggestions toward the future research in the area.

5.2 Securing IoT Architectures: Challenges and Methodology

There are many challenges when designing IoT architectures, and the most important of them stem from the fact that IoT systems are assumed to consist of very large heterogeneous networks of both constrained and unconstrained devices, continuously operating mostly without any power source. From this, one can assume that the main factors that prohibit the inclusion of strong security and privacy mechanisms in an IoT system are [6, 7]:

- the heterogeneity of the involved systems and devices in terms of communication technologies, software, hardware, and capabilities,
- the constrained nature of many IoT devices, which can have very limited resources,
- the need for scalable solutions to function properly even at large-scale deployments and
- the need for energy efficient optimization both in terms of hardware and software so that the devices can operate without the need for battery replacement for long periods.

These challenges can pose severe difficulties when designing the architecture to be secure and privacy preserving, and especially when trying to embed security and privacy functionalities on resource-constrained devices. A recent report by Hewlett Packard Enterprise [8] analysed the vulnerabilities of existing IoT hardware devices and the findings were very concerning. The highlights of this report was that more than 90 % of the devices were collecting personal information, 70 % of them used unencrypted traffic, while 60 % of them used weak credentials. It is obvious that no matter how secure the backbone IoT system is, the lack of security on the devices can really compromise the whole system, and this has to be seriously taken into account in the design of secure IoT architectures.

The IoT system architectures are designed to be secure to minimize the risks of attacks or failures. Actually, there can be either human or non-human risk sources in an IoT system. The human risk sources stem from the fact that malicious users may be hacking the devices and steal information or when users' faults or accidents affect the system performance. The non-human risk occurs when there are security issues due to natural phenomena, i.e., a flood, a fire, heat or device hardware failure.

Usually, system designers consider only human risk sources and mainly intentional malicious attacks, considering all other risk sources as out of the scope. However, using IT technologies, the impact of many other risks can be mitigated too. For example, when the reliability of the data is calculated in order to discard erroneous measurements, the data gathered by a device that is affected by a failure or a fire can be identified as erroneous and not considered in the system decisions. Furthermore, when devices are affected by flood and are not sending measurements, proper monitoring mechanisms can create alerts so that the system operator can resolve the issue and revive the devices.

As described in [9, 10] toward designing a secure IoT architecture, as a first step, one has to identify the elements (or assets) he wants to protect in an IoT system. In general, the main elements to be protected can be split into IT-based and non-IT:

- IT-based elements: this category includes assets like the user/device credentials, the various types of data that are exchanged within the system (user data, sensed or actuation data, application data, control data) and the software [9].
- Non-IT elements: this category is more generic and includes elements, such as: human users, devices (leaf or intermediary), users' privacy, services, communication channel, and in general the infrastructure [10].

For the identification of risks in IoT systems and the assessment of their impact, standard methodologies such as Microsoft's STRIDE/DREAD [11] or analysis of the Confidentiality, Integrity Availability on the assets and the threats against Authentication, Authorization and Accounting, as well as the Privacy threats. For a detailed analysis of the risks and their assessment in IoT systems, the reader can refer to [9, 10].

After the identification of the risks, the next step is to identify what are the system requirements in order to be able to mitigate these risks, and especially which are the design choices that the operator or system designer has to make in order to secure and protect his system and his data. In some cases, the system designer has to take tough but important decisions, since security/privacy and functionality/flexibility/interoperability/openness can be conflicting requirements. For example, one of the key enemies of privacy is linkability of the data, which is basically required for improved interoperability of an IoT system. Moreover, strong security comes at the expense of system performance due to the high complexity for running, i.e., powerful encryption mechanisms on the devices. However, the level of security, privacy and performance can be dynamic, selected at the operational phase so that it can correspond to the device/system capabilities and the requirements of the applications that are provided by the system. The latter is a very important requirement that has to be taken into account, since the applications can have very different security and performance requirements, and a proper IoT system has to be able to adapt to these diverse requirements. For example, an environmental monitoring application may have very low security and performance requirements, while a smart health application will have strong security, privacy, and performance requirements. Thus, the IoT systems due to the requirement for interoperability and for breaking the silos must be smart enough to be able to support both these applications without any issues.

5.3 Overview of Secure IoT Architectures

This section provides a brief analysis of the most important EU projects in the area of IoT that have embedded in their architectures security, privacy, and trust functionalities. The literature review starts with the IoT-A project, which is the lighthouse project of the EU with regards to setting the foundations for the design of IoT architectures and continues with other key projects.

5.3.1 IoT-A

The goal of the Internet of Things—Architecture (IoT-A)[1] project was to set the foundations for the design of an Architectural Reference Model (ARM) for the IoT. The project aimed to create a coherent architecture that allows the integration of heterogeneous technologies and supports the interoperability of IoT systems. IoT-A outlined principles and guidelines for the technical design of IoT communication and service provisioning protocols, interfaces, and algorithms. Furthermore, IoT-A proposed an innovative service resolution infrastructure for scalable discovery of resources and entities of the real world.

IoT-A's activities were based on the concept that the key aspect of any IoT system are the "things" and the "communication" among them. On the contrary to many past approaches that used the term "things" to denote the sensor devices, IoT-A defined the "things" to be the Physical Entities (PEs) that are all around us and are being accessed through the devices. The things that are connected to the internet are, thus, i.e., the room, a pen, a fridge, a car, a city, a laptop, anything that is a physical object and is of interest for the user. These PEs are transformed into Virtual Entities (VEs) so that they can be searchable, locatable, and controllable via software. The concept of virtualization helps to conceal the heterogeneity of the devices and to hide the devices from the applications' point of view, so that the users only request data for a PE and should not have to know which devices are monitoring or controlling this PE. This service-oriented-architecture approach utilized Resources on the IoT devices that are exposed via Services that can be reusable by many applications or can be composed to provide more complex services [10].

With regards to security and privacy, IoT-A has defined many system requirements both functional and nonfunctional that assisted in the design of the ARM. The requirements were split into three main categories, as described in [12], and summarized below:

[1] www.iot-a.eu.

- System Dependability: this category includes requirements for improving the overall security of the IoT system and its infrastructure. It includes requirements for
  - Service and infrastructure availability, so that the user can invoke a service under all conditions and the infrastructure should be able to provide this service at all times.
  - Accountability, so that any failures or misbehaviours can be traced back to the responsible person/system.
  - Infrastructure Integrity and Trust, so that the provided infrastructure services are trustworthy and can operate according to their design requirements.
  - Non repudiation so that all services can be accessed by their rightful owners.
- Communication Stack: this category includes requirements for (i) the network layer and (ii) the service layer:
  - Network layer: here the requirements are related with (i) anonymization at the network level, so that users can protect their privacy through anonymity and (ii) confidentiality, so that the messages are encrypted to be protected against eavesdroppers.
  - Service layer: here the requirements are related with (i) service authentication and access control, so that only authenticated users can have access to the system services and even more to only specific services that they are allowed to, and (ii) service trust and reputation, so that only authenticated users access the system and only trusted nodes can provide measurements.
- User and service privacy: this category includes requirements either for protecting users when they are using the Services and the Infrastructure or the users cannot extract information about the data subject that is providing a specific service.

To address the previous requirements, IoT-A identified a number of security components that are required to be included in the architecture. The architecture of IoT-A, called ARM and the respective Functionality Groups (FGs) can be seen in Fig. 8.2 in [10]. Since the focus of this chapter is only for the security and privacy components of the IoT architectures, here we will be limited to the description of the respective Security and Management FGs, while the description of the rest of the FGs is omitted.

The Management FG includes among others functionalities that can improve the resiliency and the availability of the overall system. For example, the component for "Fault" can identify and correct system faults, detecting them by generating alarms and applying corrective mechanisms. The alarms can also be sent to other components that have to act in order to correct a specific fault. Another component called "Member" handles the membership and the associated information of all relevant entities of the system. It includes a database that stores information regarding ownership, rules, rights, etc., and is important for the Security FG in order to define the security and privacy policies [10, 13].

The Security FG is the main group of functionalities for ensuring the security and privacy of the system, including the following components [10, 13]:

- **Authorisation**: which handles the security policies and takes decisions for access control based on these policies. It determines if a user action is allowed or not and controls the policies for the services by adding, updating, or deleting policies according to Service Level Agreements or user preferences.
- **Authentication**: which involves authentication both for users and services. It checks the credentials and if they are valid then it allows the access to the IoT services.
- **Identity management**: which tackles privacy issues, by distributing and managing pseudonyms and accessory information to trusted subjects to operate anonymously.
- **Key exchange and management**: which provides mechanisms for secure communications between two or more system nodes. It is also responsible for distributing keys for the secure communications in a secure way, as well as for registering security capabilities.
- **Trust and Reputation architecture**: this component collects and evaluates user reputation scores to use them for calculating service trust levels. This is done by requesting reputation information from users with respect to a service and providing the reputation of the service to interested applications or users.

Overall, IoT-A set the foundations for the architecture of IoT systems and in the proposed ARM, some basic functionalities for security and privacy where included. These functionalities can be characterized as the bare minimum that an IoT system can include so that it can be acknowledged as secure.

5.3.2 iCore

The project "Empowering IoT through Cognitive Technologies" (iCore)[2] has as a vision to provide the IoT world with the necessary technological foundations to empower the IoT with cognitive technologies, so that IoT services and applications can be easily and simply widespread. iCore structured this vision toward tackling two main issues: (i) how to abstract technological heterogeneity that is inherent to the real world objects, considering the large numbers of diverse objects that impose challenges for reliability, energy efficiency and context-awareness and (ii) how to consider the requirements of different users and stakeholders aiming to support business integrity and application provision according to service level agreements [14].

Toward these objectives, iCore structured a cognitive architectural framework that includes three levels of functionality, which can be reusable to diverse

[2] www.iot-icore.eu.

applications: (i) the Virtual Objects (VOs), (ii) the Composite Virtual Objects (CVOs), and (iii) Service Level. The VOs are used to tackle the technological heterogeneity, by enabling a cognitive virtual representation of both real world (e.g., a chair, a table, a house) and digital objects (e.g., a sensor, an actuator, a device). CVOs are cognitive mashups of semantically interoperable VOs and use the services of the latter in accordance with the user or stakeholder requirements [14].

With regards to security and privacy, iCore defined the respective requirements considering existing regulatory frameworks (i.e., EU directives) and the three main use cases of the project. As a result of the analysis of these sources, five main security requirements were defined as described in [15] and summarized below:

- **Availability**, ensuring the reliability of the data sent to authorized users.
- **Confidentiality**, ensuring the privacy of the users and the protection of their data, disallowing the disclosure of information to unauthorized individuals, devices, or services. This requirement included also the requirements for anonymization and pseudonymisation.
- **Integrity**, guaranteeing the correctness of the operation of the system according to some predefined rules and the consistency of the data.
- **Authentication or authorization**, ensuring the validity of the provision of data/services verifying the authorization of an individual to receive this information.
- **Nonrepudiation**, reassuring both the sender and the receiver of information that it was sent by the correct sender and that it was received by the proper receiver.

Considering these requirements, iCore defined also some higher level requirements for the design of the architecture [15]. These requirements were related with security functions for (i) security management, (ii) scalability and (iii) multi-level security. The goal was to ensure that security mechanisms and policies are not static and can be updated on a regular basis, to avoid using obsolete credentials or hacked software. Furthermore, the system security mechanisms have to be scalable to ensure that they can perform well in heterogeneous large-scale environments. Moreover, iCore users are considered to have multiple different levels of access, credentials, and data protection, and the framework should be able to properly handle these diverse functionalities.

The iCore Architecture presented in Fig. 2 in [14] aims to differentiate the project by allowing the system to derive knowledge from the usage of context, so that applications can reach their goals and intelligently behave and organize the system's own resources. A main concept of iCore is cognition through "virtualization," which allows operations such as (i) **functional enrichment**, targeting a virtualization with highest fidelity of managed real-world objects that coexist with virtual functional spaces, (ii) **abstraction**, allowing the generalization of the functionalities do that they are applied to a larger set of situations and requirements, and (iii) **aggregation**, allowing the combination of Virtual Objects with dynamic context-awareness capabilities so that they provide higher level functionalities, close to real world demands.

To provide enhanced system security, in the proposed iCore architecture, there is a functionality group for Security Management, which includes a policy repository, an identity provider and a Policy Decision Point (PDP) [16]. At all layers (VO, CV, service) there are Policy Enforcement Point (PEP) components to intercept any request for new service, or for executing VOs or CVOs. These requests are then sent to the PDP that evaluates the requests against predefined security policies, returning the results to the PEPs. That way proper authentication and access control is enforced in an iCore system. These functionalities are provided in iCore through a Model-based Security Toolkit (SecKit) that provides the basis for security engineering, data protection and privacy. It has meta-models that support the modeling of data, identities, context, trust, risks, and policy rules.

The iCore security architecture can provide functionalities for [16]:

- Trust management, using security policies that consider trust relationships established by users with devices and operators.
- Dynamic context adaptation, so that any context changes will be considered in the security actions.
- Data privacy, through
  - data anonymization, to allow the privacy rules to ensure proper data anonymization when they are collected by the devices,
  - control of the data flow from devices, to allow users to define which and what type of data will be gathered by the devices.
- Control of the actions of the actuators, to improve the system safety.

5.3.3 BUTLER

The BUTLER project[3] aims to support the development of pervasive applications using heterogeneous devices, protocols, and standards. The applications are built in order to improve the daily activities of users in various domains, considering among others contextual information, that may be the user needs and preferences, their location or the status of the physical entities with which the users interact. The architecture of BUTLER was built on existing standards and industry initiatives such as IoT-A and FI-WARE [17]. However, the BUTLER architecture includes additional functionalities that relate with the association of Context to virtual entities.

With regards to security and privacy, BUTLER, similarly like iCore, focused on context-aware security, adapting its mechanisms using input from its environment with regard to any changes that might occur, with the target to use this infornation to prevent inappropriate behaviors [18]. Access control mechanisms based on

[3] www.iot-butler.eu.

context were developed, for deciding if a request for access to a resource or to some data will be granted of refused. BUTLER considered security policies based on context, so that the security responses of the IoT system are adapted to their context of use. These policies have to describe the authorized practices for a user/entity of the system at each moment in time and in each situation. Furthermore, these policies are not used only for access control, but also for communication encryption and other security functionalities. Apart from context-based security, BUTLER focused also on enhancing the users' privacy [19]. BUTLER considers that the applications may impose privacy issues in term of using the data gathered by the devices.

The BUTLER architecture is depicted in Fig. 4 in [17] where the four main layers can be seen:

- The communications layer, including all functionalities for enabling the communication between heterogeneous devices and between the various entities of the system and the users. In addition, it includes various Security Services for device and user authentication and authorization to ensure that only identified and authorized devices can join the BUTLER network.
- The data/context management layer, which handles all data and context related functionalities, (e.g., capturing and collecting data, persistent storage, data processing, context extraction, etc.) but does not include any security services.
- The System/Device management layer deals with the management and the maintenance of large numbers of heterogeneous networked devices. This layer has direct interfaces with the security services, i.e., for ensuring the secure configuration of the devices, their authorized management, etc.
- The Services Layer is responsible for describing, discovering, binding, and providing context-aware BUTLER services. It is also responsible for making the services available to the applications, providing enablers for supporting discovery and purchasing of data functionalities.

As it can be seen, the main security and privacy functionalities of BUTLER reside at the communication layer, which provides access to the users to all the other functionalities. The BUTLER security services are built around a strong authorization server. The main security services are [17]:

- Secure transport of messages between devices and the authorization server, based on Transport Layer Security (TLS).
- User and application authentication, using credentials.
- Resource registration and authentication, including information on which actions are allowed to be performed on this Resource by its consumers.
- Key management, both for encryption and authorization keys.
- End to end security between application user and Resource provider.
- Device authentication via a bootstrapping mechanism between the devices and the gateways at the sensor network level, using asymmetric cryptography and elliptic curves.

As it is obvious, the main focus of BUTLER is ensuring security and privacy at the communication layer, with many functionalities built around a powerful authentication and authorization framework. This is very important in order to ensure that access to services and private information can be allowed only to authorized persons, avoiding the disclosure of information to third parties.

5.3.4 OPENIoT

OpenIoT[4] focused to develop an open source IoT middleware for getting information from heterogeneous sensor networks, concealing the types of devices that provide this information. OpenIoT adopts the IoT-A concepts of "entities" and resources, such as sensors, actuators, and smart devices, leveraging them to build the concept of "Sensing-as-a-Service", via an adaptive middleware framework for deploying and providing services in cloud environments [20].

The OpenIoT architecture is built based on various nonfunctional requirements for improved performance, scalability, reliability, privacy, and security. The project acknowledges that the provided platform has to be secure and privacy preserving. In this respect, mechanisms for role-based authentication and authorization are developed, upon which utility-driven privacy mechanisms are provided [20].

The OpenIoT architectural components mapped to IoT-A ARM are depicted in Fig. 10 in [20]. This architecture includes, among others, a group of security components, providing mainly authorization, identity management, and authentication functionalities. As described in [21], the OpenIoT platform consists of several standalone applications, such as X-GSN, LSM, etc., which require their own security mechanisms. Due to the distributed nature of the platform, OpenIoT acknowledged the importance of designing and developing a centralized authorization server that would provide authentication and authorization services to all components of the system. OpenIoT acknowledged that a main feature of the platform is that the user credentials are only checked and maintained by this central server and then, this server performs the authorization of the applications that are running on behalf of the user. In this respect, user credentials are not being circulated among the various components, which improves the system privacy.

With respect to communication security, mechanisms like TSL or HTTPS are used for the backbone communications (between the gateways and the cloud servers). At the sensor layer, standard security mechanisms of IEEE 802.15.4 are used. For the authorization framework, OAuth is used, because it is an open standard that can be easily integrated in the open platform. Apart from these, OpenIoT developed also a framework to evaluate the trustworthiness of the sensor readings via spatial correlation of the sensed measurements. In this respect, readings from neighbor sensors are correlated to assess their trustworthiness and discarding untrusted readings [21].

[4] www.openiot.eu.

5.3.5 SMARTIE

SMARTIE[5] (Secure and Smarter Cities Data Management) aims to create a framework for IoT applications that are sharing large volumes of heterogeneous data. The focus is on providing end-to-end security and information trust, considering also requirements for maintaining user's privacy. Mechanisms and technologies for security and trust at the perception, service, and network layers, as well as for secure storage and access control are central to the system.

The design of the SMARTIE system architecture was based on a long list of requirements as presented in [22]. With regards to security and privacy, SMARTIE supports the processing information on trusted nodes and user anonymity for privacy protection, data confidentiality, strong access control and authentication, secure storage, user's consent, context-aware policies, location privacy, communication integrity, no tampering of devices, and system resilience.

The SMARTIE architecture is based on the IoT-A ARM, following the IoT-A methodology and the ARM functional groups to organize all the functional components of the SMARTE architecture. The architecture is depicted in Fig. 4 in [23]. As it can be seen in this figure, there are many security related functionalities, not only in the security FG, but also embedded in other FGs. For example, the "IDS Data Distribution" component scans network traffic for intruders and reports unwanted or unknown traffic to the network operator, distributing detected events to other devices. Furthermore, the "Resource Directory with Secure Storage" component provides secure access to available resources in the system, allowing also their secure storage. The "Privacy-preserving Geofencing" component offers secure location-based services using secure geofencing, aiming to avoid the disclosure of location information of the users to unauthorized persons.

In the Security FG, SMARTIE provides the following components and functionalities [23]:

- Authorization, based on DCapBAC, for ensuring that access control decisions are taken before a service is accessed. Attribute-based access control[6] using the XACML/JSON[7] framework is used, utilizing policies to express rich and fine-grained access control decisions.
- System integrity based on nodes attestation (IMASC). IMASC provides a trusted environment for most embedded devices using SmartCards, allowing the detection of malicious code or wrong measurements.
- Authentication using distributed Kerberos and symmetric cryptography.
- Encryption, based on the CP-ABE [24] schema that ciphers information based on policies.

[5] http://www.smartie-project.eu.

[6] http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf.

[7] http://docs.oasis-open.org/xacml/xacml-json-http/v1.0/xacml-json-http-v1.0.html

- Libraries for shortECC and LmRNG, in order to provide elliptic curves with key lengths between 32 and 64 bits and to generate cryptographic secure pseudo-random numbers respectively.

SMARTIE follows also the concept of IoT-A, splitting the IoT devices in constrained and unconstrained ones. In terms of functionalities, SMARTIE proposes in [23] a recommended set of functionalities and technologies that can be used for either constrained or unconstrained devices, justifying the proposals to make sure that only lightweight implementations of technologies are embedded on constrained devices.

5.3.6 COSMOS

The COSMOS[8] project aims to enhance the sustainability of IoT-based smart city applications, enabling things to evolve and become more autonomous, reliable, and smart. The basic concept is that things will have the opportunity to learn based on the experiences of other things via a situational knowledge acquisition and analysis. Furthermore, COSMOS aims to provide end-to-end security and privacy, with hardware-coded security and privacy on storage, introducing the concept of Privetlets for IoT services.

The COSMOS architectural framework is built to address a large number of security, privacy, and trust requirements [25]. COSMOS assumes that data must be secured against eavesdroppers, data modification attacks, identity thefts or replay attacks. It also assumes that there is secure storage on the devices to protect secret information, that the devices are booted or updated in a secure way. COSMOS splits the execution environment of the services in two parts: (i) the secure part that executes the critical services and (ii) the unsecure part for the noncritical services. COSMOS uses a secure backbone server for key management, storage, and distribution, as well as for device enrollment. Authentication and access control are also embedded in the system.

The notion of Trust is also very important for COSMOS. A Trust Model for providing data integrity and confidentiality is defined, allowing also endpoint authentication and nonrepudiation between any system entities. Furthermore, COSMOS considers the notion of Privacy as very important for protecting the private information of users. Privacy is assumed to be supported by enabling entities to maintain control over their private information and decide whether they will be gathered, used, or disclosed to other entities [25].

COSMOS defined security mechanisms across different layers. The project aimed to ensure end-to-end security and privacy with security embedded at the device level, access control, encryption and cross-application mechanisms on the data level, injecting privacy-preserving mechanisms whenever appropriate. For the

[8] www.iot-cosmos.eu.

definition of the COSMOS architecture, the IoT-A ARM was used as a basis. The functional view of the architecture is depicted in Fig. 7 in [26].

As depicted in this figure, the main security components of the architecture reside in the Security FG

- **Authentication and authorization** is used for authenticating entities and managing the allocation of their access rights.
- **Key exchange and Management** covers the generation, storage, and distribution of cryptographic keys, based on the Diffie and Hellman [27] key exchange protocol.
- **Hardware Board Communication Accountability** handles the issue of accountability, tracking access to crypto primitives for nonrepudiation purpose, computing the reputation index of the entity.
- **Cryptographic nonrepudiation** is enforced based on the hardware board communication accountability component.
- **Checksum** ensures the integrity of the data packets, detecting, and correcting bit errors.

In the rest of the FGs there are also other components that are related to security, privacy, and trust

- **Privetlets** are acting as filters to ensure that every virtual entity and every user shares only the minimum intended information, omitting all other unnecessary information in order to maintain the privacy.
- **Communication channel** between VEs or between VEs and the COSMOS system has to be secured, thus encryption of the data transferred within these channels is applied.

In general, COSMOS gives much importance on ensuring privacy via data minimization at the middleware layer, as well as on strong authentication and authorization. Encryption at the communication channel is also important for ensuring confidentiality of the data.

5.3.7 COMPOSE

The COMPOSE[9] project developed an IoT platform that eases the task of developers writing applications which are based on the Internet of Things (IoT). Following section is based on the information from the project's deliverables [28, 47] and [29]. COMPOSE abstracts from the IoT devices by assigning to them a virtual identity. Devices which are not directly connected to the Internet (e.g., a bottle of wine with a RFID or NFC tag) will need a proxy to represent them in the COMPOSE platform. IoT objects with network capabilities, but without support of the

[9] http://www.compose-project.eu.

network protocols needed for COMPOSE, such as simple sensors, will also use proxies to be able to communicate with the COMPOSE platform. Finally, there is a group of advanced devices (so-called Smart Objects, such as a Smart Phone, tablet, or an Arduino device) that are capable to communicate with the COMPOSE platform directly. All the above-mentioned physical objects are called Web Objects in COMPOSE and are represented as Service Objects. COMPOSE specifies an API by which it expects to communicate with the Web Objects, in order to obtain data from them, or set data within them (for more detailed information see [45]). A Service Object exhibits a standard API also internally towards the rest of the components within the COMPOSE platform. That API is needed in order to streamline and standardize internal access to Service Objects, which can in turn represent a variety of very different Web Objects providing very different capabilities.

The COMPOSE platform, in an effort to embrace as many IoT transports as possible, allows Web Objects to interact with their representatives in the Platform (the Service Objects) using a set of well-known protocols: HTTP, STOMP over TCP, STOMP over WebSockets, and MQTT over TCP. Out of Service Objects, COMPOSE offers to design applications. Developers locate interesting existing Service Objects or applications, and tailor-specific logic around them. In addition, Service recommendation will be made available to choose the best suitable entity based on developer needs as well as proposed composition, and recommendation based on platform knowledge, such as security related aspects.

The security architecture of COMPOSE is based on the approach of data-centric security requirements [29]. It differs from classical device-centric approaches in that the COMPOSE security framework shrinks the security perimeter to the granularity of data. It is fine granular and can be combined with static and dynamic enforcement to regain governance on devices and data without sacrificing the intrinsic openness of IoT platforms.

The framework is depicted in Fig. 5.1 and it encompasses the following concepts:

- **Security metadata** is stored together with the entities they refer to. Metadata captures security policies of users specifying the privacy level the system must maintain for them. Service-centric metadata also allows developers and providers to specify the use of the services or service objects. Finally, the metadata will be consulted to efficiently build secure applications and workflows, and to store provenance information for data generated and processed in COMPOSE, and to store reputation information about users, service objects, and COMPOSE applications.
- **Provenance information** is solely generated by the COMPOSE system. It archives the information about when, who, and how an entity has been used. It accumulates information about the applications generating specific data, the applications consuming it, and possible operations performed on this data, e.g., its combination with other data or its broadcasting to remote locations outside of the COMPOSE system. To gather precise provenance information runtime monitoring is performed during sensor update dispatching and execution of

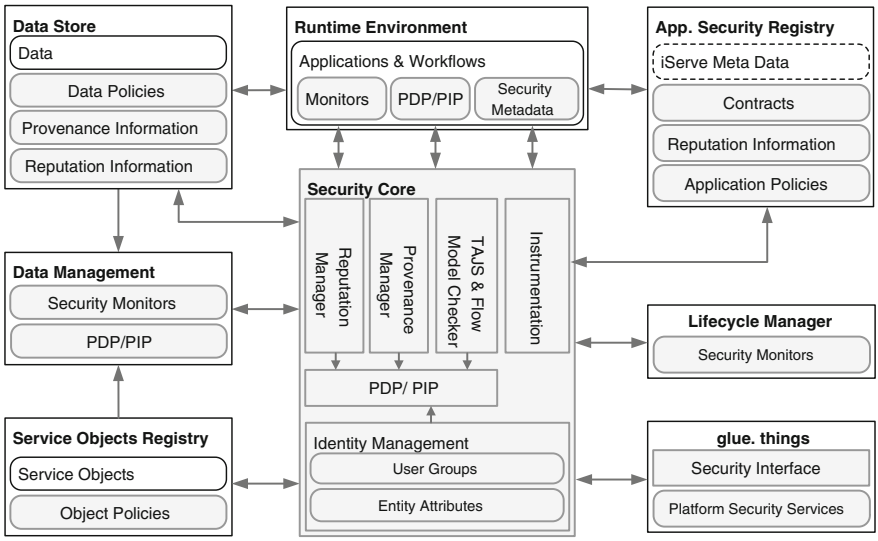

Fig. 5.1 COMPOSE security architecture overview [47]

applications. The data provenance manager tracks origins of data, the operations performed on it, and the time when operations took place. This empowers users to define policies based on data provenance, e.g., allow a Service Object to receive data only if it has been processed by a particular application. Further, visualizing the provenance of data can help users to detect when certain errors occur; for example, if several data sources are combined, but one of them is malfunctioning, the developer could examine the sources of correct values, and compare them with wrong values to isolate the malfunctioning device. Also, provenance information has an interesting potential to help to protect the user's privacy. For instance, it could eventually help to detect when particular applications harvest and correlate information from specific entities, hinting to the possibility of user profiling.

- **Reputation information** stores in the corresponding service object and service registries the collected feedback about service objects, and COMPOSE applications. Reputation information can be collected regarding service objects but also about the COMPOSE application popularity. The component called PopularIoTy[10] reflects how often a certain Service Object or application is used, i.e., invoked by other entities. Whenever data is generated, notifications are stored in the data storage. Moreover, monitors are placed in the runtime to store

[10]Parra, J.D.: PopularIoTy: http://github.com/nopbyte/popularioty-api/ (2014) and PopularIoTy Analytics: http://github.com/nopbyte/popularioty-analytics/ (2014).

notifications when an application is called. All this information is processed to calculate a popularity score. It also interacts with the contracts for applications, such that a contract compliant application will get rewarded with a positive score.

- **Contracts for applications** define conditions for applications stating when they can be executed securely, as well as flow specifications which is information about their internal, abstract data flows. While conditions specify system states which must hold before execution of an applications or which hold after its execution respectively, flow specifications provide more insights about where, e.g., to which resource, input data is flowing, and how and from where output data is generated. This information can be generated by hand describing critical security services provided by COMPOSE, e.g., the encryption or authentication of a data stream. Once an API will be published, a semiautomated process will generate contracts for these APIs. This information is used in a deployed COMPOSE environment to enable and simplify the analysis process. In general, i.e., for user-defined applications, contracts are generated by the static analysis component of the security core to save computational resources during the analysis of COMPOSE workflows.
- **Data flow enforcement** is enforced dynamically and statically. A data-centric flow policy is attached to input and output data for all components inside COMPOSE. Apart from specifying the entities allowed to access, execute, or alter a component, flow policies also describe the security requirements of data entering a component and the security properties of data leaving it. A unified policy framework can be used to avoid additional evaluation overhead, i.e., in COMPOSE the language is inspired by ParaLocks [30].
- **Identity management** is provided through a generic attribute-based IDM framework. It associates COMPOSE entities with identifiers and stores and distributes appropriate security information to also provide an authentication service. In COMPOSE an attribute-based approach allows every user to tag himself or his entities with attribute values. Once entities are tagged with attributes, e.g., the brand of the device, the tag can be used to specify security policies, e.g., accept data only from devices from that brand. COMPOSE allows users to approve attribute information depending on the group where they belong [31].
- Finally, **enforcement points** at the runtime level enforce access to data, services, or other resources based on decisions of the policy decision point (PDP). COMPOSE at its core embraces information flow security, runtime monitors (part of the PDP) detect and prevent illegal flows—as specified by the user or service object provider—during the execution of user-provided services. Finally, the PDP guides the security analysis and instrumentation components in COMPOSE whose task is the identification of potentially malicious flows in applications or workflows and their prevention or mitigation by software reconfiguration or instrumentation.

5.3.8 PRISMACLOUD

The PRISMACLOUD[11] project is a EU-funded research project developing the cryptographic tools to build more secure and privacy-preserving cloud services. To **enable end-to-end security** for cloud users and provide tools to **protect their privacy** the project brings novel cryptographic concepts and methods to practical application. The following information is based on the already available project deliverables [32, 33]. PRISMACLOUD wants to increase the pervasion of already existing and maybe nearly usable strong cryptographic primitives to the practice. It was perceived as being low at the beginning of the project and thus identified as an obstacle that withholds the use of less-trusted intermediaries (like cloud provided services, IoT gateways, or IoT middleware) in many security and privacy conscious usage scenarios, e.g., Smart Cities. PRISMACLOUD aims at bridging the divide between the needed deep cryptographic knowledge and the application requirements of cloud users in order to bring the cryptographic primitives to good practical use.

PRISMACLOUD's architecture encapsulates the cryptographic knowledge of the primitives layer inside the tools and their usage inside services. As depicted in Fig. 5.2, it is organized into four tiers. Each layer helps abstracting from the needed core cryptographic primitives and protocols, which are located at the Primitives layer, which is the lowest layer of the PRISMACLOUD architecture. Building the tools, the layer above, from the primitives requires in depth cryptographic and software development knowledge. However, once built they can be used by cloud service designers to build cryptographically secure and privacy-preserving cloud services. Thus, PRISMACLOUD's architecture levels also define connection points between the different disciplines involved: cryptographers, software engineers/developers, and cloud service architects. On the uppermost (i) Application layer are the end-user applications. Applications use the cloud services of the (ii) Services layer to achieve the desired security functionalities. The cloud services specified there are a representative selection of possible services, which can be built from the tools organized in the (iii) Tools layer. In particular, they represent a way to deliver the tools to service developers and cloud architects in an accessible and scalable way. Together the tools constitute the PRISMACLOUD toolbox.

The PRISMACLOUD architecture can be seen as one recipe to bring cryptographic primitives and protocols into cloud services such that they empower cloud users to build more secure and more privacy-preserving cloud services. In particular, it considers providing tools for secure object storage, flexible authentication with selective disclosure, verifiable data processing, topology certification, and data privacy. The services designed in PRISMACLOUD are data sharing; secure archiving; selective authentic exchange; privacy enhancing ID management; verifiable statistics; infrastructure auditing; encryption proxy; anonymization service. The tools are using the following cryptographic primitives: RDC: Remote Data

[11] http://www.prismacloud.eu.

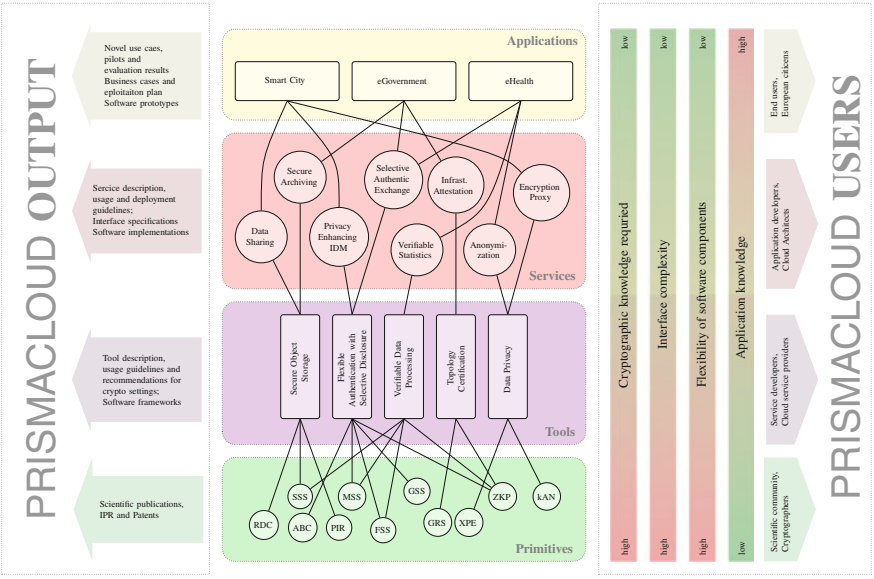

Fig. 5.2 Overview of PRISMACLOUD's 4-tier architecture [33]

Checking; SSS: Secret Sharing Schemes; ABC: Attribute-Based Credentials; PIR: Private Information Retrieval; MSS: Malleable Signature Schemes; FSS: Functional Signature Schemes; GSS: Group Signature Schemes; GRS: Graph Signature Schemes; SPE: Format- and Order-Preserving Encryption; ZKP: Zero-Knowledge Proofs; kAN: k-Anonymity (abbreviations from Fig. 5.2).

To take it to a Smart Cities example, imagine a number of IoT devices that constantly record data, but have no storage and computing capabilities to do higher level aggregation. Assuming further that this aggregation will be done on a cloud-based infrastructure then this infrastructure must be trusted to perform the computation correctly. Here the trust in this infrastructure can be removed using a cryptographic primitive called functional signatures. They allow the delegation of signature generation to other parties for a class of messages meeting certain conditions. Such schemes can be used to certify the computation done by third parties, such as untrusted intermediaries like the gateway or the middleware of an IoT stack. From this class of signature schemes PRISMACLOUD builds the Verifiable Data Processing Tool. The tool allows performing verifiable computations on data such as computing statistics on IoT data. See [32] for more information on cryptographic signature primitives that PRISMACLOUD uses.

5.3.9 RERUM

RERUM[12] aims to develop an IoT architectural framework adopting the concepts of "security and privacy by design". RERUM acknowledges that an IoT system cannot be fully secure post-development but has to be designed from its foundations to be secure and privacy preserving. This is the main difference of RERUM when compared with other architectural approaches, such as the ones presented above. RERUM's architecture design process followed the acknowledged methodology of IoT-A, but additionally, RERUM put a significant focus on the IoT devices. This was done because RERUM acknowledges the fact that up to now, the weakest point in an IoT system was the constrained devices, which did not have the capabilities to run advanced security and privacy mechanisms. Indeed, this lack of security focus on devices resulted in many open security and privacy holes which were not limited only on the devices and but expanded to the overall system [34, 35].

Apart from the nonfunctional requirements for "security and privacy by design" RERUM considered also the requirements for increased system reliability, robustness, resilience, and availability to ensure that the system can respond to attacks and that the data will be available to be provided to the applications whenever they are requested. Since RERUM's key focus is on the devices, the key differentiating factor of its functional requirements is that the developed security and privacy mechanisms were required to be lightweight and energy efficient so that they can be easily implemented and embedded even on constrained IoT devices. RERUM's requirements include among others, lightweight encryption, confidentiality, and integrity protection, authorized modification of integrity protected data, simple and strong authentication both for users and devices, attribute-based access control, user consent, data collection limitation, data minimization, accountability, secure device bootstrapping, and secure configuration of devices [36].

The RERUM architecture went through a detailed process of requirements definition and analysis in order to extract the required functional components to support the long list of requirements. RERUM defined its own functionality groups called "managers". Among others, RERUM defined the "Security, Privacy and Trust Manager" (SPT) that included a long list of functional components. These components are depicted in Fig. 5.3 mapped to the IoT-A ARM's FGs [37].

What is interesting from this mapping is that contrary to previous IoT security architectures, RERUM developed a large number of components that are assumed to be embedded on the IoT devices. For example, components for secure credential bootstrapping, secure storage, geolocation privacy enhancing techniques, integrity generator/verifier, data encryption/decryption, trusted routing, cognitive radio security, and device to device authentication are assumed to be important to be running on the devices to ensure the highest level of security of an IoT system, by strongly securing the leaf nodes.

[12] www.ict-rerum.eu.

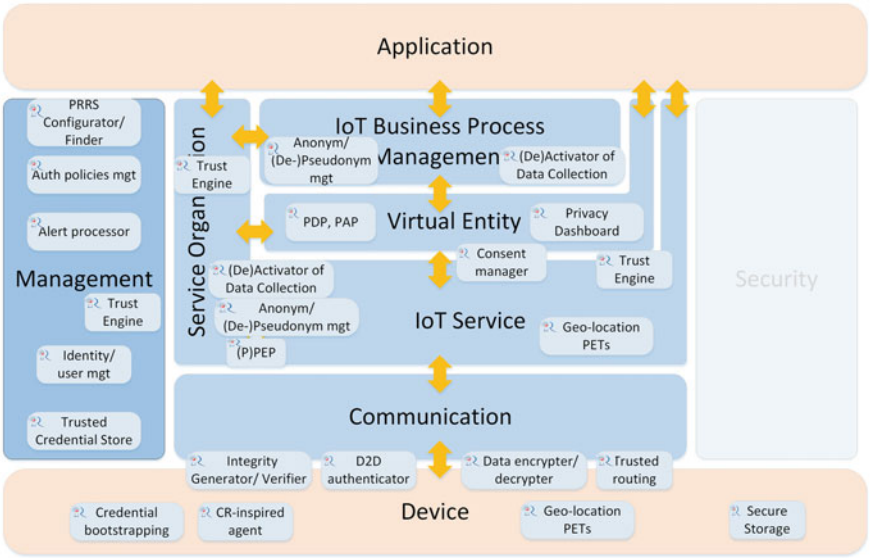

Fig. 5.3 RERUM architectural security components mapped on the IoT-A ARM [37]

Apart from that, security components for authentication and access controlled are also important in RERUM, together with secure reconfiguration of devices, creating and processing security alerts, handling user and device identities and storing securely credentials and certificates. A number of components are also defined for ensuring user privacy via anonymization and pseudonymisation, activation and deactivation of data collection based on user preferences, a consent manager to request the consent of the user for gathering or disclosing private information, a privacy dashboard to allow users themselves to handle their privacy policies, as well as Policy Decision and Enforcement Points to manage and execute the policies.

Trust is also very important in RERUM. The project has defined a trust engine to calculate and evaluate the trust ratings for devices, services and users, based on observers scattered around different system entities and which are monitoring the user actions, the data reliability as these are gathered by the devices, and the behavior of the devices in order to identify malicious or misbehaving devices so that these will not affect system's decisions. That way, the reliability of the overall system is improved.

5.4 Conclusions

The Internet of Things is becoming an important element of the everyday lives of the people, providing opportunities for simplified activities, and improved quality of life. Acknowledging the numerous benefits of the IoT for the people, the

environment and the economy, it has attracted a lot of interest from the research community. However, until recently, the focus was given towards enabling the provision of advanced services to the users, with only limited attention towards security and privacy. Nevertheless, lately a number of key contributions from EU-funded projects have changed the IoT research landscape, with important advancements in the domains of security, privacy, and trust.

Having the reference framework of IoT-A as the foundation, many EU projects have designed their architectures with strong features of security and privacy, each one aimed to address their specific objectives. As it was analysed above, all projects have added functionalities for authentication and authorization/access control in order to ensure that the services of the system will be provided only to those users that are allowed to access them and to none else. This is achieved through the security and privacy policies that define the roles of each user and the actions he can perform. However, the proposed concept of context-awareness in the access control allows the policies to adapt to situational changes, i.e., a doctor will be allowed to enter an apartment if the inhabitant had a heart attack.

Cryptography in IoT has also been considered by most projects in various forms, i.e., for key and identity management so that proper keys and identities are provided to the various entities of the system. Encryption has also been supported by many projects, mainly though on the backbone communications, since the IoT devices can be very constrained to support encrypted communications. In this respect, the DTLS technology using ECC can play a significant role [38] or the novel idea of Compressive Sensing-based encryption, which performs simultaneous compression and encryption in the measurements, contributing to the minimization of the energy consumption of the devices [39–41].

Data integrity is also a very important issue in IoT, since malicious or tampered data can severely affect the decisions of the system and could potentially even harm people when actuators are involved. To ensuring data integrity, techniques such as digital signatures creation and verification must run on the devices [42]. This way the receiver can validate that no unauthorized intermediate node has tampered with the data it received. If subsequent modifications are needed, advanced techniques like malleable signatures can be applied to allow changes to parts of the data limited to authorized nodes for concealing identifiable information without breaking the validity of the signature [32, 43].

Data integrity can also be a metric of the reliability of the data, which is used for measuring the trustworthiness of the IoT system. Not many projects have worked towards trust and reputation schemes for the devices, but the focus was mainly to assess the reputation of users in order to change their access policies.

With regards to privacy, the main functionalities developed by most projects were related with the inclusion of privacy policies on the access control mechanisms. Only a few projects lately have worked towards cross-layer privacy enhancing techniques for data minimization, for ensuring unlinkability of the data, for allowing users to take control of their data and for anonymization and pseudonymisation of the data so that no identifiable information is disclosed to unauthorized third parties. Location privacy is also a key research item in some

projects and this has mainly been tackled via geofencing/data minimization and anonymisation/pseudonymisation.

Overall, there have been significant advances in the areas of security and privacy in IoT in the last few years. However, as also analysed in [44], there are many solutions that are not addressed adequately so far in the IoT community. For example, embedding autonomic computing functionalities in the security mechanisms for implementing self-management based security is not addressed so far, although it can contribute to a more resilient IoT system. Research towards unobservable communications for improved privacy is also a very interesting topic, with the goal to try to avoid extracting the measurements from performing traffic analysis [4]. Traffic anonymization in IoT networks, using, i.e., Tor or I2P is yet an open research area. Anonymous credentials and identity mixers can highly contribute to improved privacy protection. Finally, trust building between devices using trust negotiation protocols requires the iterative exchange of credentials and strong cryptographic calculations, which is not applicable to constrained IoT devices. These are only some examples of open research items that can improve even more the security and privacy of IoT systems. However, the main challenge is the design of the respective functionalities in a lightweight and energy efficient way, so that they can be embedded on resource constrained devices. Only then, the IoT systems can be fully secure and the citizens will have the necessary incentives to adopt this exciting new set of technologies.

Acknowledgements This work has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreements no 609094, 612361 and 644962.

References

1. Evans D (2011) The internet of things. How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG)
2. Ranken, Margaret, M2M Global Forecast & Analysis 2014–24, Machina Research Strategy Report, 24, June 2015
3. Ruiz D et al (eds) (2015) Enhancing the autonomous smart objects and the overall system security of IoT based Smart Cities, RERUM Project Deliverable D3.1, 28 February 2015
4. Pöhls HC et al (eds) (2015) Privacy enhancing techniques in the Smart City applications, RERUM Project Deliverable D3.2, 2 Sept 2015
5. Blefari-Melazzi N, Bianchi G, Salgarelli L (eds) (2011) Trustworthy internet. Springer Science & Business Media
6. Vermesan O, Friess P (2014) Internet of things—from research and innovation to market deployment. River Publishers
7. Vermesan O, Friess P (eds) (2015) Building the hyperconnected society: IoT research and innovation value chains, ecosystems and markets, vol. 43. River Publishers
8. Internet of Things research study. Hewlett Packard Enterprise 2015 report. www8.hp.com/h20195/V2/GetPDF.aspx/4AA5–4759ENW.pdf
9. Mouroutis T et al. (eds) (2014) Use-cases definition and threat analysis, RERUM Project Deliverable D2.1, 31 May 2014

10. Bassi A et al (2013) Enabling things to talk. Designing IoT solutions with the IoT architectural reference model, pp 163–211
11. Howard M, Lipner S (2006) The security development lifecycle: SDL: a process for developing demonstrably more secure software. Microsoft Press (2006)
12. Gruschka N et al (eds) (2012) Concepts and Solutions for Privacy and Security in the Resolution Infrastructure, IoT-A Project Deliverable D4.2, 16 February 2012
13. Carrez F et al (eds) (2013) Final architectural reference model for the IoT v3.0, IoT-A Project Deliverable D1.5, 15 July 2013
14. Menoret S et al (eds) (2014) Final architectural reference model, iCore Project Deliverable D2.5, 2 Nov 2014
15. Baldini G et al (eds) Security requirements for the iCore cognitive management and control framework, iCore Project Deliverable D2.2, 31 May 2012
16. Neisse R, Steri G, Fovino IN, Baldini G (2015) SecKit: a model-based security toolkit for the internet of things. Comput Secur 54:60–76. ISSN 0167-4048
17. Integrated System Architecture and Initial Pervasive BUTLER proof of concept, BUTLER Project Deliverable D3.2, October 2013
18. Requirements, Specifications and Security Technologies for IoT Context-Aware Networks, BUTLER Project Deliverable D2.1, October 2012
19. Ethics, Privacy and Data Protection in BUTLER, BUTLER Project Deliverable D1.4, July 2013
20. Dimitropoulos P, Soldatos J, Kefalakis N, Bengtsson JE, Giuliano A et al (eds) OpenIoT detailed architecture and proof-of-concept specifications, OpenIoT Project Deliverable D2.3, 28 March 2013
21. Gwadera R et al (eds) (2013) Privacy and Security Framework, OpenIoT Project Deliverable D5.2.1, 27 Sept 2013
22. Azevedo R et al (eds) (2014) Requirements, SMARTIE Project Deliverable D2.2
23. Skarmeta A et al (eds) (2015) Initial Architecture Specification, SMARTIE Project Deliverable D2.3
24. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: IEEE symposium on security and privacy. SP'07, pp 321–334
25. Pitu L et al (eds) (2015) End-to-End Security and Privacy: Design and Open Specification (Updated), COSMOS Project Deliverable D3.1.2, 30 April 2015
26. Carrez F et al (eds) (2015) Conceptual Model and Reference Architecture (Updated), COSMOS Project Deliverable D2.3.2, 30 April 2015
27. Diffie W, Hellman ME (1976) New directions in cryptography. IEEE Trans Inform Theory 22 (6):644–654
28. Deliverable D1.2.2 Final COMPOSE architecture document
29. Schreckling D, Parra JD, Doukas C, Posegga J (2015) Data-Centric Security for the IoT. In: Proceedings of the 2nd EAI international conference on IoT as a Service, Rome, Italy
30. Broberg N, Sands D (2010) Paralocks: Role-based information flow control and be- yond. In: Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on principles of programming languages. pp 431–444. POPL'10, ACM, New York, NY, USA
31. Parra J, Schreckling D, Posegga J (2014) Identity Management in Platforms Of fering IoT as a Service. In: 1st international conference on IoT as a service. Lecture Notes in Computer Science (LNCS), Springer, Rome, Italy
32. Demirel D, Derler D, Hanser C, Pöhls HC, Slamanig D, Traverso G (2015) PRISMACLOUD D4.4: Overview of Functional and Malleable Signature Schemes
33. Lorünser T, Länger T, Slamanig D, Pöhls HC (2016) PRISMACLOUD Tools: A Cryptographic Toolbox for Increasing Security in Cloud Services. In: Proceedings of a workshop on security, privacy, and identity management in the cloud collocated at the 11th international conference on availability, reliability and security. ARES'16, Salzburg, Austria, 2016

34. Pohls HC et al (2014) RERUM: Building a reliable IoT upon privacy-and security-enabled smart objects. In: Wireless communications and networking conference workshops (WCNCW), 2014 IEEE. IEEE, 2014
35. Tragos EZ et al (2014) Enabling reliable and secure IoT-based smart city applications. In: 2014 IEEE international conference on pervasive computing and communications workshops (PERCOM Workshops). IEEE, 2014
36. Cuellar J et al (eds) (2014) System Requirements and Smart Objects Model, RERUM Project Deliverable D2.2, 31 May 2014
37. Tragos E et al (eds) (2015) Final System Architecture, RERUM Project Deliverable D2.5, 4 Sept 2015
38. Capossele A et al (2015) Security as a CoAP resource: an optimized DTLS implementation for the IoT. In: IEEE international conference on communications (ICC), IEEE, 2015
39. Charalampidis P, Fragkiadakis A, Tragos E (2015) Rate-adaptive compressive sensing for IoT applications, VTC2015-Spring, Glasgow
40. Fragkiadakis A, Tragos E, Papadakis S, Charalampidis P (2014) Experiences with deploying Compressive Sensing and Matrix Completion techniques in IoT devices, IEEE CAMAD 2014, Athens, 2014
41. Fragkiadakis A, Tragos E, Traganitis A (2014) Lightweight and secure encryption using channel measurements, Wireless Vitae 2014, Aalborg
42. Pöhls HC (2015) JSON Sensor Signatures (JSS): End-to-End Integrity Protection from Constrained Device to IoT Application. In: 9th international conference on innovative mobile and internet services in ubiquitous computing (IMIS). IEEE, Santa Cantarina, Brazil, 2015
43. Pöhls HC, Samelin K (2015) Accountable redactable signatures. In: 10th international conference on availability, reliability and security (ARES). IEEE, 2015
44. Baldini G et al (2015) Internet of Things. IoT Governance, Privacy and Security Issues, European Research Cluster on The Internet of things, Activity Chain 05 Whitepaper, January 2015
45. Trifa V, Larizgoitia I (2013) Design of the object virtualization specification, Compose Deliverable D2.1.1, 30 Oct 2013
46. Fragkiadakis A, Angelakis V, Tragos EZ (2014) Securing cognitive wireless sensor networks: a survey. Int J Distrib Sens Netw (2014)
47. Schreckling D et al (2015) The Compose Security Framework, COMPOSE Deliverable D5.4.2, 15 Nov 2015