Chapter 7

Pattern-driven Security, Privacy, Dependability and Interoperability in IoT

By Nikolaos Petroulakis, Konstantinos Fysarakis, Henrich C. Pöhls, Vivek Kulkarni, George Spanoudakis, Arne Bröring, Manos Papoutsakis, Manolis Michalodimitrakis and Sotiris Ioannidis

Copyright © 2020 Nikolaos Petroulakis *et al.*
DOI: 10.1561/9781680836837.ch7

The work will be available online open access and governed by the Creative Commons "Attribution-Non Commercial" License (CC BY-NC), according to https://creativecommons.org/licenses/by-nc/4.0/

Published in *Security Risk Management for the Internet of Things: Technologies and Techniques for IoT Security, Privacy and Data Protection* by John Soldatos (ed.). 2020. ISBN 978-1-68083-682-0. E-ISBN 978-1-68083-683-7.

Suggested citation: Nikolaos Petroulakis *et al.*. 2020. "Pattern-driven Security, Privacy, Dependability and Interoperability in IoT" in *Security Risk Management for the Internet of Things: Technologies and Techniques for IoT Security, Privacy and Data Protection*. Edited by John Soldatos. pp. 121–142. Now Publishers. DOI: 10.1561/9781680836837.ch7.

now
the essence of knowledge

This chapter presents the development of a pattern-driven approach for guaranteeing Security, Privacy, Dependability and Interoperability (SPDI) properties in the IoT domain. The chapter details how SPDI patterns can be introduced to guarantee multi-layer end-to-end properties, and how the enforcement of the patterns can help and satisfy the requirements for network dependability guarantees. Moreover, it briefly evaluates the presented approach and thus demonstrates how the application of patterns offers a solution to semantic interoperability challenges in IoT environments.

7.1 Introduction

While the fifth generation (5G) of mobile communications is already dawning upon us, the next steps in their evolution will be key in supporting this societal transformation, while also leading to a fourth industrial revolution that will impact multiple sectors. Next-generation networks, such as the Internet of Things (IoT), aim to create open and global networks for connecting smart objects, network elements, applications, web services, and end users. Research and industry

attempt to integrate this evolving technology and the exponential growth of IoT by overcoming significant hurdles such as dynamicity, scalability, heterogeneity, and end-to-end security and privacy. The introduction of digital technologies in economic and societal processes is key to addressing economic and societal challenges such as aging of population, ensuring societal cohesion, and sustainable development. IoT appears to be an important pillar of 5G. Global networks like IoT create enormous potential for new generations of IoT applications, by leveraging synergies arising through the convergence of consumer, business, and industrial internet, and creating open, global networks connecting people, data, and *things*. A series of innovations across the IoT landscape have converged to make IoT products, platforms, and devices technically and economically feasible. However, despite these advances, significant business and technical hurdles must be overcome before the IoT's potential can be realized.

Some important challenges and complexities include:

- Sustaining massively generated, ever-increasing, network traffic with heterogeneous requirements
- Adaptation of communication technologies for resource-constrained virtualized environments
- Provision of networking infrastructures featuring end-to-end connectivity, security, and resource self-configuration
- Trusted information sharing between tenants and host systems.

Overcoming these challenges requires the implementation and deployment stack of IoT applications. The overall aim of SEMIoTICS[1] proposes the development of a pattern-driven framework [1, 2], built upon existing IoT platforms, to enable and guarantee secure and dependable actuation and semi-autonomic behavior in IoT and Industrial Internet of Things (IIoT) applications. The SEMIoTICS framework supports cross-layer intelligent dynamic adaptation, including heterogeneous smart objects, networks, and clouds. To address the complexity and scalability needs within horizontal and vertical domains, SEMIoTICS develops and integrates smart programmable networking and semantic interoperability mechanisms.

The SEMIoTICS architectural framework (Figure 7.1) has been envisaged and developed for efficient interconnectivity of smart objects. Each layer contains specific developed modules able to handle different aspects and guarantee different properties. More specifically, Software Defined Networking (SDN) Orchestration layer provides data and control plane decoupling resulting in a cloud computing

1. SEMIoTICS: Smart End-to-end Massive IoT Interoperability, Connectivity and Security: http://www.semiotics-project.eu

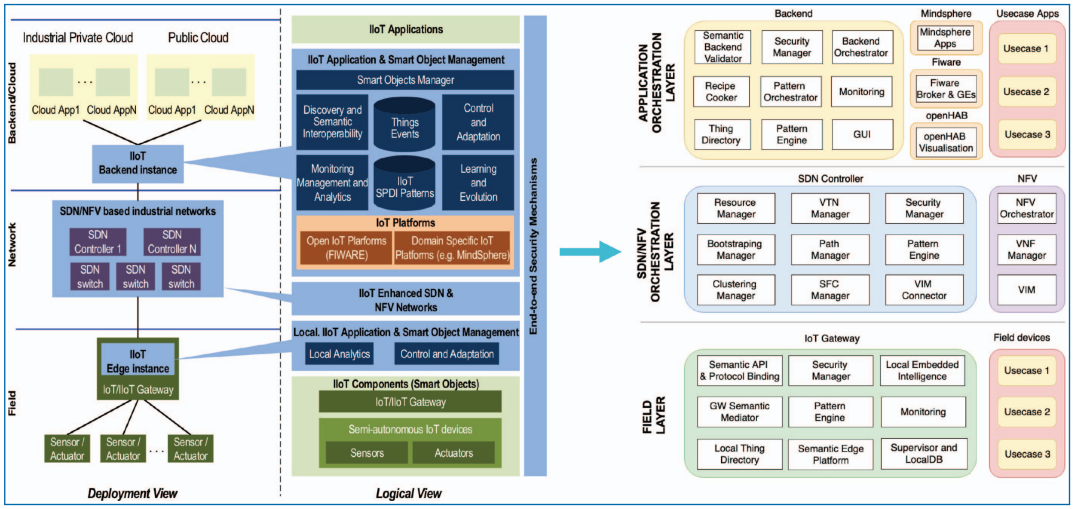


Figure 7.1. SEMIoTICS envisaged and developed architecture.

approach that facilitates network management and enables programmatically efficient network configuration meeting different IoT application requirements related to security, bandwidth, latency, and energy efficiency, using semantic information. Network Function Virtualization (NFV) Orchestration layer provides a flexible, programmable, dynamic, and scalable networking paradigm, making it ideal for satisfying the Quality of Service (QoS) demands of SEMIoTICS use cases. Field layer is responsible for hosting all types of IoT devices such as sensors and actuators as well as IoT gateway, which provides common way for communication and ensures enforcement of SPDI patterns in this layer. Finally, Application Orchestration layer consists of all applications receiving the communication from field layer.

This chapter is organized as follows: Section 7.2 presents the IoT challenges and the background of security, privacy, dependability and interoperability (SPDI)-related issues. The main concept and the key building blocks of the proposed pattern-driven IoT service orchestration solution are detailed in Section 7.3. Section 7.4 provides a glimpse on the proposed SPDI patterns. Finally, Section 7.5 features the concluding remarks and pointers to future work.

7.2 Background and Challenges

The background of the SPDI properties and the necessity for the usage of patterns to drive the fulfillment of the end users' goals in the directions of security, privacy, dependability and interoperability for the IoT are provided in this section. Furthermore, the IoT challenges stemming from each of the four domains are also briefly mentioned in order to highlight that being able to address them system-wide and in a *by-design-fashion* is of paramount importance in industrial or larger IoT environments.

7.2.1 IoT Security

Smart objects, IoT applications, and their enabling platforms are often vulnerable to security attacks and changing operating and context conditions that can compromise their security [3]. This is exacerbated by the fact that they typically generate, make use of, and inter-relate massive personal data in ways that can potentially breach legal and privacy requirements. Preserving security and privacy properties remains a particularly challenging problem, due to the difficulty of: (a) analyzing vulnerabilities in the complex end-to-end compositions of heterogeneous smart objects; (b) selecting appropriate controls for smart objects with heterogeneous resources/constraints (e.g., different schemes for ID and key management, different encryption mechanisms); and (c) preserving end-to-end security and privacy under dynamic changes in IoT applications and security incidents, in the context of the ever-evolving IoT threat landscape [4]. In this context, basic security tasks such as mutual authentication, encryption, and data integrity remain challenging in IoT, with a variety of lightweight solutions proposed [5]. Confidentiality and integrity protection mechanisms also require strong authentication and authorization mechanisms. Existing solutions requiring user involvement [6] or certificates [7] lack scalability and support for dynamic IIoT networks. Security and privacy at the IoT back-end, e.g., the cloud and centralized servers, requires distribution of data and processing, which necessitates the use of new distributed and/or collaborative paradigms of cloud computing [8]. To ensure that the most sensitive private data remain secure from source to end user and only accessible to authorized entities, one of the solutions is to encrypt data based on policies for access control, in addition to using secure communication channels. Thus, specific research areas such as Attribute-based Encryption (ABE) need to be improved in order to make them more efficient and scalable [9].

7.2.2 Privacy Invasion

With an increased consumer awareness by privacy invasions on general media or on industrial IoT applications, the need for privacy protection becomes immanent. From a business perspective, the problem of privacy comes framed as the problem to protect companies trade secrets. Additionally, privacy legislation can be a major concern in certain markets more than in others. Lately, the legal concerns have been fueled by heightened fines for privacy violations introduced in data protection laws, especially Europe's GDPR [10, 11]. The GDPR [10] requires among other things that personal data may be gathered only for a precisely specified purpose. It also requires to minimize the amount of collected data [12] and the data subject. Moreover, the individual person who's personal data are handled needs to give their informed consent a priori to the data gathering and must be able to intervene.

Further to being a legal or business challenge, privacy could also be seen as a human right in either way privacy matters in the IoT domain and as such we discuss some privacy challenges which have to be taken care of at the architectural level:

- **Handle IoT devices as first-class citizens:** Unsuspicious information communicated within the IoT could too easily be used by attackers to extract companies trade secrets or infringe the human right to protect personal information. Privacy requires to tackle the privacy problems already at the field level, i.e., already deploy sensors and actuators which are capable of tackling privacy problems [13]. As a minimum step, this requires to protect confidentiality of any data that is communicated, i.e., by encrypting traffic [14].
- **Minimize transferred data:** Data minimization can be achieved in many ways, by simple aggregation of data, but SEMIoTICS is even smarter and performs intelligent analytics locally [15].
- **Provision of privacy enhancing mechanisms and services** Required to implement in practical IoT deployments the technological advances that allow to provide data with certain quality guarantees while still containing less private information [16–18].
- **Specify privacy requirements:** A device that is communicating might leak, i.e., even encrypted traffic may still be vulnerable to being analyzed for length of packets, frequency of communication, and other observable communication meta-data. Thus, achieving unobservability of communication [19] would be the highest goal for IoT privacy [20]. To achieve privacy holistically and deploy the right mechanisms correctly, privacy must become part of the engineering framework [21].
- **Monitor potential privacy breaches:** The patterns approach allows to address and specify privacy requirements. But, the potential of design patterns for privacy goes beyond a single policy language cause; they allow communication between different actors in different domains. Finally, they are ideal for enabling information privacy into information systems [22].

The latter mentioned challenge of monitoring allows the data subject to exercise some form of oversight, which adds to the transparency of the data processing.

7.2.3 Network Dependability

Dependability is the ability of a system to deliver its intended level of service to its users [23]. The main attributes which constitute dependability are reliability, availability, safety, and maintainability. Dependable systems impose the necessity to provide higher fault and intrusion tolerance. The satisfaction of these attributes can avoid threats such as faults, errors, and failures offering fault prevention; fault

tolerance; and fault detection. More specifically, dependability in SEMIoTICS is focused on three major attributes such as reliability, availability, and fault tolerance as follows:

- **Reliability** is the ability of a system to perform a required function under stated conditions for a specified period of time [24]. It is an attribute of system dependability, and it is also correlated with availability. For hardware components, the property is usually provided by the manufacturer. This is calculated based on the complexity and the age of the component. Reliability can be classified into two main categories: the deterministic models and the probabilistic ones.
- **Availability** guarantees that information is available when it is needed [24]. The lack of availability in network transmissions has a severe influence on both the security and the dependability of network. More specifically, network availability is the ability of a system to be operational and accessible when required for use. Moreover, availability in networks is the probability of successful packet reception [25]. Other factors which affect the availability of a link are the transmission range of the signal strength, noise, fading effects, interference, modulation method, and frequency.
- **Fault Tolerance** is the ability of a system or component to continue normal operation despite the presence of hardware or software faults [24]. Network fault tolerance appears to be a critical topic for research [26]. The most common solutions to guarantee fault tolerance and avoid single point of failure include the replication of paths forwarding traffic in parallel, the use of redundant paths, and the ability to switch in case of failure (failover) and traffic diversity. Fault tolerance mechanisms exist in all layers of field, network, and back-end/cloud. In the field layer, failures involve the drop of sensors or actuators and the gateways. More specifically, fault tolerance in network architectures requires the design of a network able to guarantee avoidance of single or multiple link failures, faulty end hosts and switches, or attacks. The key technical solution of the problem includes the creation of a fault tolerance mechanism to provide open-flexible design where existing fault tolerance solutions are not effective.

Dependability analysis of an IoT system includes whether non-functional requirements such as availability, reliability, safety, and maintainability are preserved. The conditions depend on the respective dependability property that the system guarantee. The satisfiability of a property can be defined by a Boolean value (i.e., true, false), an arithmetic measure (i.e., delay), or probability measure (i.e., reliability/uptime availability).

7.2.4 IoT Interoperability

Interoperability gives an ability to a system or a product to connect and work with other systems or products. Interoperability is defined as *a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, present or future, in either implementation or access, without any restrictions* [27]. The following types of interoperability can be distinguished and are covered in SEMIoTICS [28]:

- **Technological interoperability** enables seamless operation and cooperation on heterogeneous devices that utilize different communication protocols. Technological interoperability still remains a significant barrier in IoT settings as up to 60% of the overall potential value is currently locked due to lack of compatible solutions [29].
- **Syntactic interoperability** can establish clearly defined formats for data, interfaces, and encoding. In terms of syntactic interoperability, IoT vendors typically claim to utilize standardized and widely used technologies and platforms in order to increase the acceptance of their products. Nevertheless, the variety of common choices, such as the Constrained Application Protocol (CoAP), eXtensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP), MQ Telemetry Transport (MQTT), Devices Profile for Web Services (DPWS), and Universal Plug and Play (UPnP), leads to a fragmented landscape [30]. Mechanisms for resolving this issue include gateway proxies for the messaging protocols [31], but again their presence cannot be assumed in all environments and their introduction leads to increased complexity in interactions and additional costs.
- **Semantic interoperability** settles commonly agreed information models and ontologies for the used terms that are processed by the interfaces or are included in exchanged data. It is materialized by including information regarding the data (metadata) and linking each element to a commonly shared vocabulary. As IoT integrates an extremely large amount of heterogeneous entities, these entities need to be consistently and formally represented and managed. The Open Geospatial Consortium (OGC) [32] and Semantic Web Activity of the World Wide Web Consortium (W3C) [33] help provide enhanced descriptions and meaning to sensor data. Several semantic frameworks like Semantic Sensor Network (SSN) ontology [34] and IoT-Lite [35] have been proposed in the international literature. One of the main concerns leading to poor adoption of IoT semantic descriptions is that semantic techniques increase the complexity and processing time, and therefore, they are unsuitable for dynamic and responsive environments such as the IoT.

- **Organizational interoperability** cross-domain service integration and orchestration through common semantic and programming interfaces. The common interpretation of semantic information in a globally shared ontology could be useful in enabling cross-domain and cross-organization interactions. However, this is not always the case. Although several local systems may utilize popular or standardized ontologies, eventually they extend them and establish their own semantics and interfaces. The use of Semantic Information Broker (SIBs) is proposed the literature [36], while other approaches focus on providing common and generic Application Programming Interfaces (APIs) between the different IoT middleware platforms, towards a marketplace of applications and services (e.g., BIG IoT [37] project).

7.2.5 Achieving Security, Privacy, Dependability and Interoperability by Design

Research and industry communities alike stress the need to adopt a *by design* approach as the most effective means of addressing the above challenges, i.e., considering these early from the design phase. In this context, the pattern-driven approach of SEMIoTICS follows the *security-by-design* concept, which aims to guarantee system-wide security properties by virtue of the design of the involved systems and their subsystems. This is leveraged to provide orchestration-level SPDI guarantees, while encompassing all involved components and entities which are composed to create the orchestrations (e.g., physical devices and software). A key capability required in security-by-design is the ability to verify the desired security properties as part of the design process. A typical way to achieve this is using model-based techniques [38–40], whereby software component and service compositions are modeled using formal languages, and the required security properties are expressed as properties on the model [41]. The satisfiability of the required properties is based on model checking [42, 43]. Other approaches focus on software service workflows using business process modeling languages (e.g., Sec-MoSC [44]). Pino *et al.* [45] use Secure Service Orchestration (SSO) patterns to support the design of service workflows with required security properties, leveraging pattern-based analysis to verify security properties. This avoids full model checking that is computationally expensive and non-scalable to larger systems, such as the IoT. Moreover, some model-based approaches (e.g., [45]) support the transformation of security requirements to code for automated checking of the required properties, both at design and at run time.

The SEMIoTICS pattern-driven framework's operation is inspired by the many works that have successfully used design patterns as a mean to communicate across

different stakeholder domains and to reuse proven solutions for problems which occur over and over again. The initial works date back to the 1977 book "A Pattern Language: Towns, Buildings, Construction" by Alexander *et al.* [46], where the concept of reusable design solutions for architectural problems was introduced. The pattern approach has ever since been successfully applied in other domains, e.g., software design [47], security [48], privacy [49]. Here, SEMIoTICS, especially, builds upon ideas from similar pattern-based approaches used in service-oriented systems [50, 51], cyber-physical systems [52], and networks [53, 54], covering more aspects in addition to Security, and especially privacy [55, 56] and also providing guarantees and verification capabilities that span both the service orchestration and deployment perspectives, as detailed in Section 7.3 above.

7.3 SPDI Patterns

To enable the pattern-driven approach, it is necessary to develop a language for specifying the components that constitute IoT applications along with their interfaces and interactions. In this context, the definition of the various functional and non-functional properties of IoT components and their orchestrations is required in the form of a model. The defined model appears in Figure 7.2 and is presented in detail in [2]. A model with such characteristics effectively serves as a general "architecture and workflow model" of the IoT application.

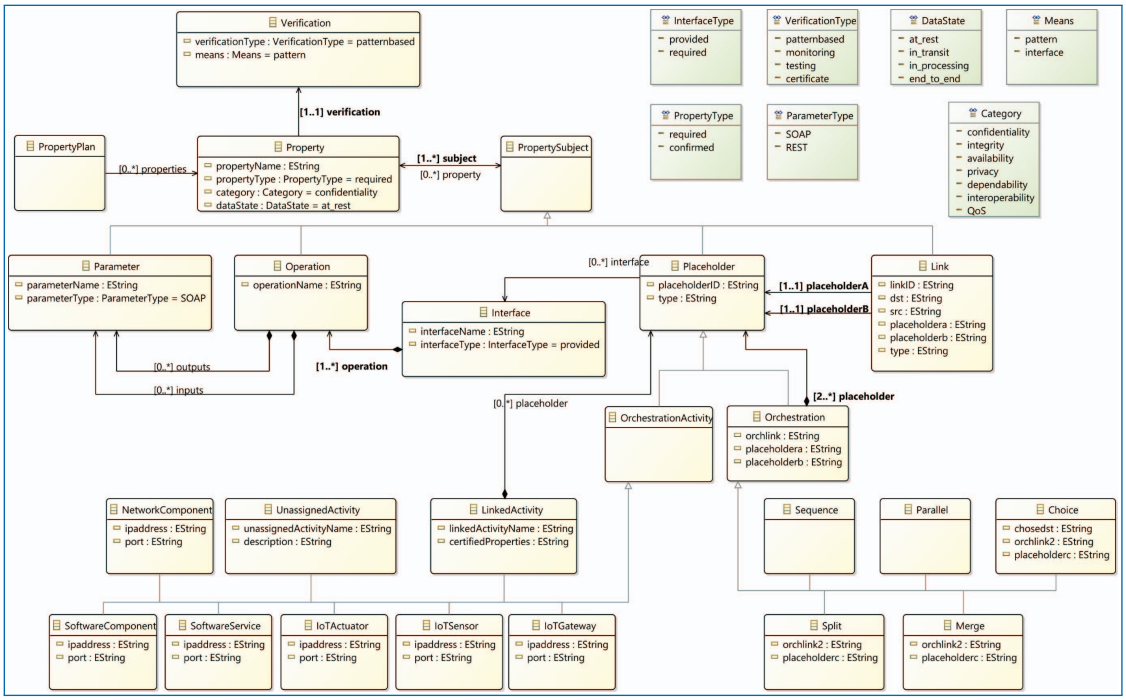


Figure 7.2. IoT orchestrations system model.

7.3.1 Pattern Language

Once defined, this model is used to derive a language which will allow the definition of pattern rules and facts which, consequently, enable the reasoning required for verifying SPDI and QoS properties in specific IoT applications and subsequently enable different types of adaptations. The derived language for defining IoT application models adopts an orchestration-based approach. An orchestration of activities may be of different types depending on the order in which the different activities involved in it must be executed (e.g., sequence, parallel, choice, merge). Moreover, an orchestration involves orchestration activities. The implementation of an activity in an IoT application orchestration may be provided by a software component, software service, network component, an IoT sensor, actuator or gateway, as well as a sub-orchestration of IoT application activities of the previous types. These types of IoT application activity implementers are grouped under the general concept of a placeholder, which is accessible through a set of interfaces.

Overall, this language: (i) provides constructs for expressing and encoding dependencies between SPDI properties at the component and at the composition/orchestration level; (ii) is structural, without prescribing exactly how the functions should be executed nor, e.g., how the ports ensure communication; (iii) supports the static and dynamic verification of SPDI properties, and; (iv) can be automatically processable by the SEMIoTICS framework so that IoT applications can be adapted at run time.

Patterns expressed in the above-defined language enable the pattern-based IoT application management process followed in SEMIoTICS, in which patterns are used to: design IoT applications that satisfy required SPDI properties; verify that existing IoT applications satisfy required SPDI properties at design time, prior to the deployment of the application; and enable the adaptation of IoT applications or partial orchestrations of components within them at run time in a manner that guarantees the satisfaction of required SPDI properties.

To fulfill the above, SPDI patterns encode proven dependencies between security, privacy, dependability and interoperability properties of individual components of IoT applications and corresponding properties of orchestrations of such components. More specifically, a pattern encodes relationships of the form:

$$P1 \wedge P2 \wedge \ldots \wedge Pn \rightarrow Pn + 1 \tag{7.1}$$

In the above, Pi (i = 1,…,n) are properties of individual components, and Pn+1 is a property of the orchestration of these components. The relation encoded by a pattern is an entailment relation.

The run-time adaptations that can be enabled by SPDI patterns may take three forms: (a) to replace particular components of an orchestration; (b) to change the structure of an orchestration; and (c) a combination of (a) and (b).

7.3.2 Machine-processable Pattern Encoding

An important requirement for implementing the SPDI pattern-driven management and adaptation of the IoT infrastructure is to support the automated processing of developed patterns. To achieve this, the SPDI patterns can be expressed as *Drools* [57] business production rules, and the associated rule engine, by applying and extending the Rete algorithm [58]. The latter is an efficient pattern-matching algorithm known to scale well for large numbers of rules and datasets of facts, thus allowing for an efficient implementation of the pattern-based reasoning process. A Drools production rule has the following generic structure:

```
rule name
<attributes>*
when <conditional element>*
then <action>*
end
```

7.3.3 Reasoning Components

Pattern-related components are present in all layers of the SEMIoTICS framework (Figure 7.3), in line and towards realizing the SEMIoTICS vision of embedded intelligence across all layers of the IoT deployment.

In more detail, these components are:

- *Pattern Orchestrator*: Module featuring an underlying semantic reasoner able to understand IoT Orchestrations and workflows, as received from the Recipe Cooker module, and transform them into composition structures (orchestrations) to be used by architectural patterns to guarantee the required properties. The Pattern Orchestrator is then responsible to pass said patterns to the corresponding Pattern Engine (as defined in the back-end, network and field layers), selecting for each of them the subset of these that refer to components under their control (e.g. passing network-specific patterns to the Pattern Engine present in the SDN controller).
- *Backend Pattern Engine*: Features the pattern engine for the SEMIoTICS back-end, along with associated subcomponents (knowledge base, reasoning engine). It is able reason on the SPDI properties of aspects pertaining to the operation of the SEMIoTICS back-end. Moreover, at run time the

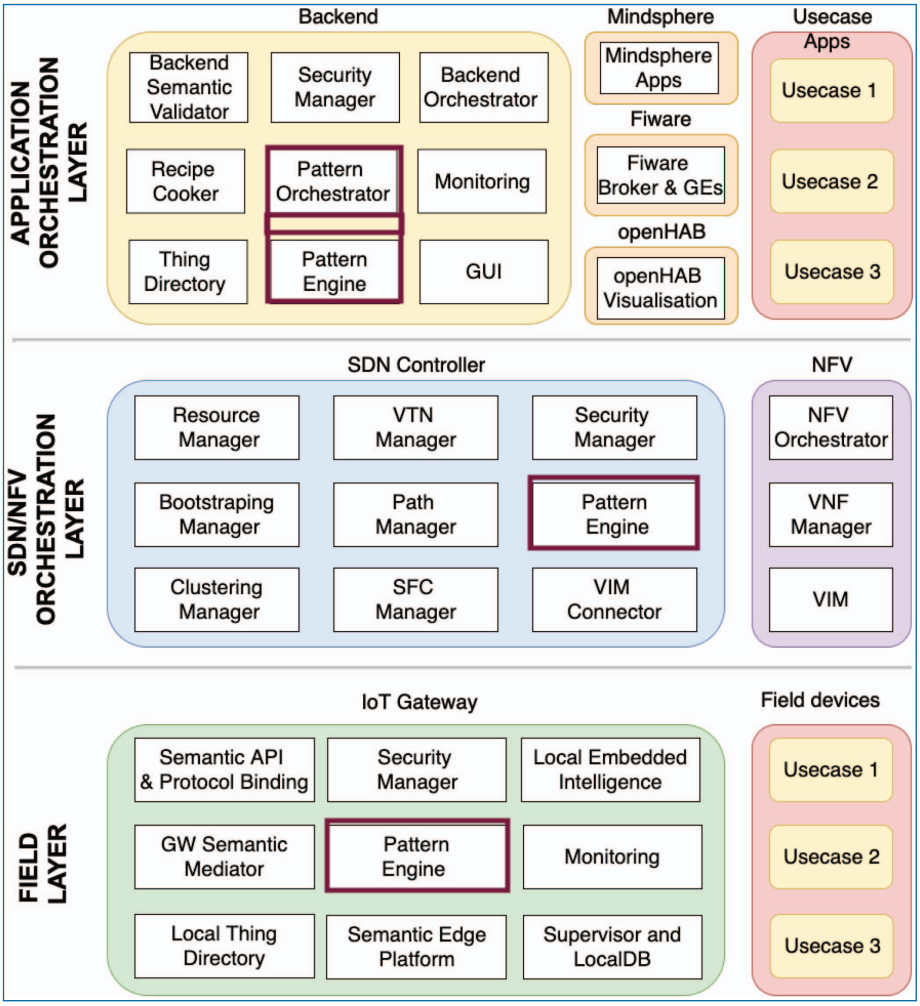


Figure 7.3. Pattern reasoning engines at all layers of the SEMIoTICS architecture.

Backend Pattern Engine may receive fact updates from the individual Pattern Engines present at the lower layers (Network & Field), allowing it to have an up-to-date view of the SPDI state of said layers and the corresponding components

- *Network Pattern Engine*: Integrated in the SDN controller to enable the capability to insert, modify, execute, and retract network-level patterns at design or at run time. It is supported by the integration of all required dependencies within the network controller, as well as the interfaces allowing entities that interact with the controller to be managed based on SPDI patterns at design and at run time.
- *Field Layer Pattern Engine*: Typically deployed on the IoT/IIoT gateway, able to host design patterns as provided by the Pattern Orchestrator. Since the compute capabilities of the gateway can be limited, the module is able to host patterns in an executable form compared to the pattern rules as provided in the other layers.

7.4 Pattern Enforcement's and Evaluation in SEMIoTICS Use Cases

Patterns in SEMIoTICS framework are enforced to satisfy SPDI properties on the different use cases. In the next subsections, a demonstrated scenario to enable pattern-driven IoT orchestration to satisfy QoS requirements is presented. Moreover, the policy enforcement through the pattern engine to monitor privacy violations is provided. Finally, the service function chaining patterns are also described to enable the classification and forward traffic through the respective security service functions in the SEMIoTICS pattern framework.

7.4.1 Pattern-enabled IoT Orchestrations

A demonstration scenario that relies on the SEMIoTICS pattern-driven network interface and its capabilities was designed and developed in industrial IoT environments and more specifically oil leakage detection in wind turbines through video monitoring. The overarching aim of the scenario is to distribute a complex application (composed of multiple tasks) to a network of IoT device and specify constraints (through patterns) on the network orchestration. In this context, the developed scenario leverages a user-friendly design and deployment of IoT orchestrations. The two key research innovation of the scenario and associated demonstration relate to: (1) True distribution of application flows over multiple devices and representing the network perspective and (2) Automated enforcement of network orchestration constraints by defining them as SEMIoTICS patterns.

In the above, other than the user-friendly, graphical interface and distributed nature of defining the IoT orchestrations involved (including where/on which devices parts of a flow are deployed), we also want to define SPDI and QoS between these deployments. Focusing on the network aspects, while maintaining the high-level abstractions needed for user-friendliness, a "Network Link" node enables direct communication between distributed Node-RED instances. Said "Network Link" node enables definition of QoS constraints (e.g., minimum bandwidth, latency) and the whole orchestration specification (a "Recipe") [59, 60], and the QoS constraints are translated into the SEMIoTICS pattern language and sent to Pattern Orchestrator [61, 62]. From the latter, the information is relayed to the network (SDN) Pattern Engine. A high-level view of this process is shown in Figure 7.4.

7.4.2 Security and Privacy Policy Enforcement Patterns

The main storyline is focused on a patient living in a smart home environment. In this environment, the patient's information is kept confidential by default;

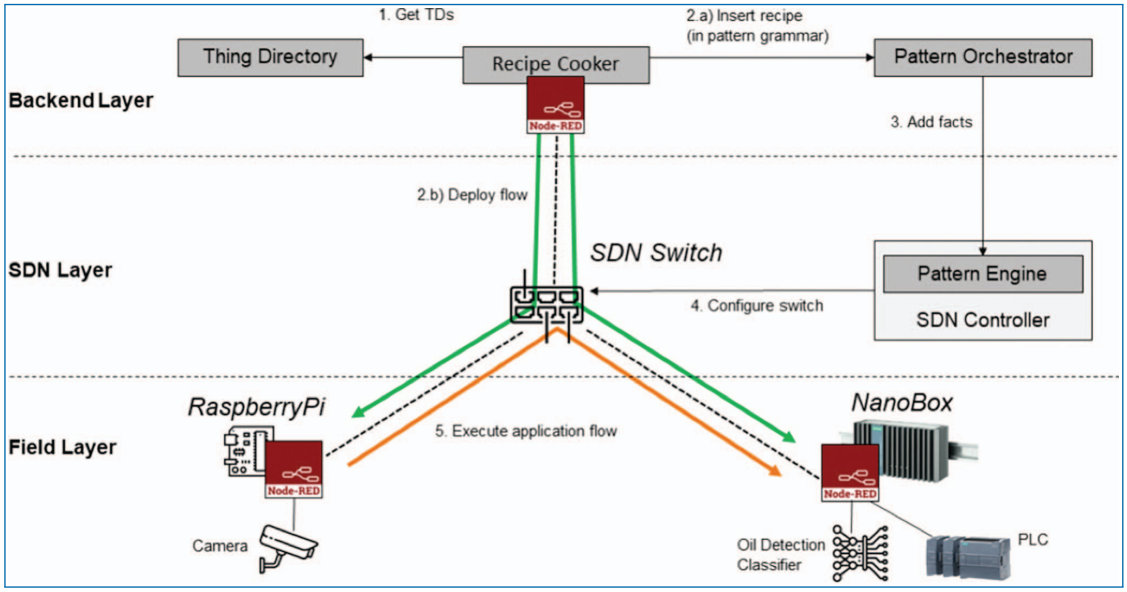


Figure 7.4. Pattern enforcement QoS [61].

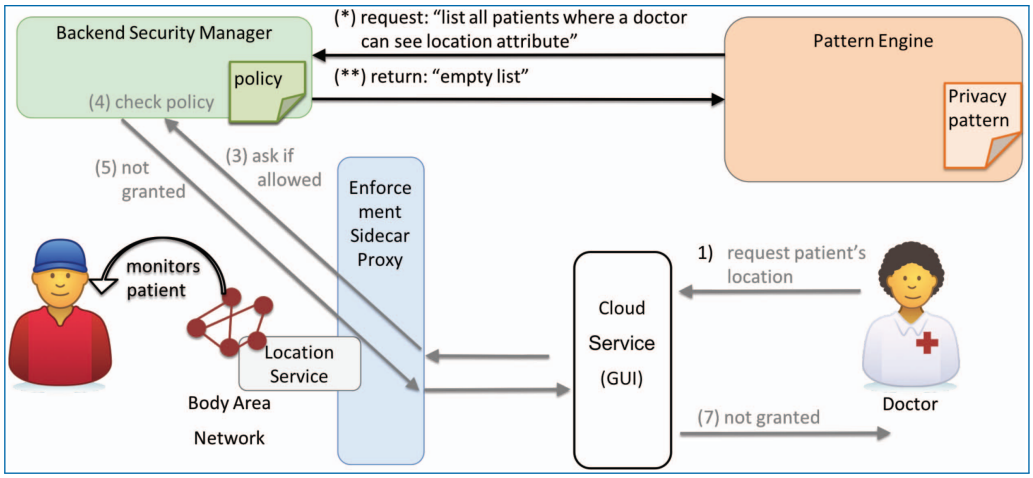


Figure 7.5. Interaction security manager and pattern engine.

however, in the event of an emergency, e.g., a patient fall, this information needs to be available to a selected set of authorized users, e.g., in this case medical personnel. The process, also depicted in Figure 7.5, includes a doctor's initial request to a service where the request is checked for validity by a *policy enforcement point* (PEP). Asking a SEMIoTICS component called *Security Manager* is only needed if the service would be exposed by the field level and after the synchronization and evaluation of the Security Manager. Based on this information, the PEP will then decide if the request or the response will be granted or not.

With respect to the scope of privacy, the SEMIoTICS *Security Manager* as depicted in Figures 7.3 and 7.5 acts to ensure that the privacy properties imposed by the pattern are technically enforced. In detail, this happens by enabling the interaction of a Security Manager located in the back-end with the Pattern Engine through the associated privacy pattern rule as depicted in Figure 7.5. The goal is that the

Pattern Engine is able to check if the current policy used to make the decisions inside the Backend Security Manager is conforming to the SPDI Patterns as specified by the application. The Pattern Engine thus periodically makes requests to the Security Manager to obtain the necessary information to judge the quality of the currently enforced policy. For example, to check that no one has access to a patient's location, it queries to obtain a list of entity that would be granted access. Based on the response, the appropriate pattern rule expressing the SPDI pattern can reason on the answers received from the Security Manager. For example, under normal health conditions of all patients that response shall be an empty list. By reasoning over the response, the Pattern Engine is able to identify if the policy being enforced in SEMIoTICS is compliant with the privacy pattern; this is done without having to check the actual technical enforcement (e.g., the access control policy as specified in the PEP), but rather by checking its effectiveness; further, it is able to reflect this to the outside via an API call that will allow other components to retrieve the SPDI status.

7.4.3 Service Function Chaining and SPDI Patterns

E-health monitoring systems situated at homes can facilitate the monitoring of patients' activities and enable the remote provision of healthcare services. They improve the quality of elder population well-being in a non-obtrusive way, allowing greater independence, maintaining good health, preventing social isolation for individuals, and delay their placement in institutions such as nursing homes and hospitals. One of the scopes of SEMIoTICS is to provide security guarantees through the traffic forwarding via different network security functions by applying the Service Function Chaining (SFC). The main focus of scenario is to support the traffic classification based on the predefined SFC for providing secure chains to forward the different kind of traffic of this use case. SEMIoTICS framework can be applied in order to support the SFC mechanism to guarantee security and dependability based on the defined SPDI patterns instantiating the required (i) Virtual Network Function (VNFs) and (ii) SFC for assuring the SPDI requirements. Traffic classification is based on the predefined SFC for providing secure chains to forward the different kinds of traffic of this use case. The procedure of instantiation and the identification of the respective SFCs and the VNFs based on the patterns are depicted in the Figure 7.6. Considering the different types of traffic reaching the back-end where the chaining of services will take place, a variety of intricacies can be observed such as of low trust and low priority, low bandwidth and latency, medium trust but high priority, medium trust and of low priority, and finally high trust and high priority, as low latency and relatively high bandwidth.

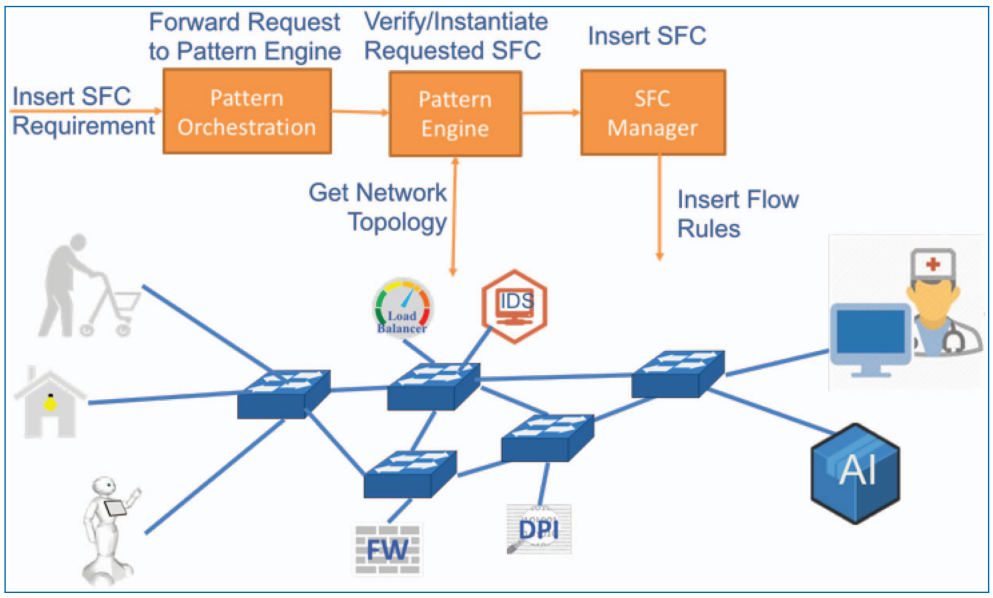


Figure 7.6. Pattern enforcement service function chains.

The design of an efficient control flow mechanism is required to be used not only to verify SFC and VNFs but also to instantiate them for assuring the SPDI requirements based on the enforcement of the respective SPDI patterns. When an SFC cannot be verified, the required VNFs are requested by the Virtual Infrastructure Manager (VIM) via NFV Orchestrator to identify them or to instantiate them if they do not exist. The procedure of instantiation and the identification of the respective SFCs and the VNFs based on the patterns are depicted in the Figure 7.6 including the following interactions with the components of the SEMIoTICS architecture. Pattern Orchestrator forwards a specific chain request to the pattern engine for forwarding the traffic between entities through a specific chain of functions. Pattern engine forwards this request to the SFC manager which is located in the SDN controller responding to the pattern engine whether the chain exist or not. If the chain exists, then a respond of the chain satisfaction is returned to the Pattern Orchestrator. If the chain does not exist, then a request is forwarded to the VIM asking whether the service functions exist or not. If functions exist in the VIM, then the chain can be instantiated in the SFC Manager and a respond of the chain satisfaction is returned to the Pattern Orchestrator. If functions do not exist in the VIM, then a function instantiation request is forwarded to the NFV Orchestrator, which is responsible to instantiate them in the VIM. Then, the chain can be instantiated in the SFC Manager, and a respond of the chain satisfaction is returned to the Pattern Orchestrator. Based on the provided dynamic instantiation of service chains and service functions through the pattern engine, the potentiality within this use case can be increased and extended by the support of additional service chains to enable traffic classification through different combinations of service functions to guarantee different secure end-to-end traffic forwarding.

7.5 Conclusion

This chapter presented a pattern-driven framework addressing the often very diverse and complex and not well-scaling requirements of commercial, societal, and industrial IoT applications. The work presented the SEMIoTICS framework's approach towards the development of patterns for orchestration of smart objects and IoT platform enablers in IoT applications with guaranteed security, privacy, dependability and interoperability properties. The definition of the pattern language and its development was also presented. Moreover, the chapter showcased the usefulness of a pattern-driven approach by describing how it works to increase the security, privacy, interoperability and dependability in very different and specific application scenarios. With this approach, the very diverse and system-spanning goals of even large-scale IoT deployments also for industrial IoT can be defined and securely and reliably orchestrated to meet the required SPDI properties.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreements No. 780315 (SEMIoTICS).

References

[1] N. E. Petroulakis *et al.*, "SEMIoTICS Architectural Framework: End-to-end Security, Connectivity and Interoperability for Industrial IoT." In 2019 IEEE Global IoT Summit (GIoTS), 2019.

[2] K. Fysarakis *et al.*, "Architectural Patterns for Secure IoT Orchestrations." In 2019 IEEE Global IoT Summit (GIoTS), 2019.

[3] M. Kert *et al.*, "State of the Art of Secure ICT Landscape." NIS Platform WG 3, V2, April 2015.

[4] ENISA, "Threat Landscape Report." https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2016, 2016.

[5] C. Manifavas, G. Hatzivasilis, K. Fysarakis, K. Rantos, "Lightweight Cryptography for Embedded Systems – A Comparative Analysis." In Data Privacy Management and Autonomous Spontaneous Security. DPM 2013, SETOP 2013. Lecture Notes in Computer Science, vol 8247. Springer, Berlin, Heidelberg, 2014.

[6] T. Marktscheffel, W. Gottschlich, W. Popp, P. Werli, S. D. Fink, A. Bilzhause and H. de Meer. "QR Code Based Mutual Authentication Protocol for Internet of Things." In Proc. of The 5th workshop on IoT-SoS: Internet of Things Smart Objects and Services (WOWMOM SOS-IOT 2016), IEEE, 2016.

[7] R. Hummen *et al.*, "Towards viable certificate-based authentication for the internet of things." In Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy, 2013.

[8] Big DataWorking Group. Expanded Top Ten Big Data Security and Privacy Challenges. Cloud Security Alliance, 2013.

[9] M. Chase and S. S. Chow. Improving privacy and security in multi-authority attribute-based encryption. Proceedings of the 16th ACM conference on Computer and communications security – CCS'09, page 121, 2009.

[10] European Parliament and the Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)." In Official Journal L 119 of 4.5.2016, pp. 1–88, 2016.

[11] J. Leyden, "Last year's ICO fines would be 79 times higher under GDPR." The Register, http://www.theregister.co.uk/2017/04/28/ico_fines_post_gdpr_analysis/, 2017.

[12] EU Article 29 Data Protection Working Party, "Opinion 8/2014 on the on Recent Developments on the Internet of Things." 2014.

[13] H. C. Pöhls *et al.*, "RERUM: Building a Reliable IoT upon Privacy- and Security- enabled Smart Objects." In Proc. of the Workshop on Internet of Things Communications and Technologies (IEEE WCNC 2014), pp. 122–127. IEEE, 2014.

[14] Z. Tragos, V. Angelakis, A. Fragkiadakis, D. Gundlegard, S. Nechifor, G. Oikonomou, H. C. Pöhls and A. Gavras, "Enabling reliable and secure IoT-based smart city applications." In Proc. of Conference on Pervasive Computing and Communications (IEEE PERCOM 2014), pages 111–116. IEEE, 2014.

[15] SEMIoTICS, "Deliverable D4.3 – Embedded Intelligence and Local Analytics." https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5c9a45e55, 2019.

[16] H. C. Pöhls and M. Karwe, "Redactable Signatures to Control the Maximum Noise for Differential Privacy in the Smart Grid." In Proc. of the 2nd Workshop on Smart Grid Security (SmartGridSec 2014), Springer, 2014.

[17] T. Lorünser, D. Slamanig, T. Länger and H. C. Pöhls. "PRISMACLOUD Tools: A Cryptographic Toolbox for Increasing Security in Cloud Services." In Proc. of the Workshop on Security, Privacy, and Identity Management in the Cloud to be held at the 11th International Conference on Availability, Reliability and Security (ARES SECPID 2016), Conference Publishing Services (CPS), August, 2016.

[18] D. von Oheimb and J. Cuellar, "Designing and Verifying Core Protocols for Location Privacy." In Proc. of International Conference on Information Security (ISC), pp. 502–516, Springer, 2006.

[19] R. C. Staudemeyer, H. C. Pöhls and M. Wojcik. "The road to privacy in IoT: beyond encryption and signatures, towards unobservable communication." In Proc. WOWMOM 2018, IEEE, July, 2018.

[20] R. C. Staudemeyer, H. C. Pöhls and M. Wojcik. "What it takes to boost Internet of Things privacy beyond encryption with unobservable communication: a survey and lessons learned from the first implementation of DC-net." In Journal of Reliable Intelligent Environments (JRIE), 5 (1), pp. 41–64, 2019.

[21] A. Kung *et al.* "A Privacy Engineering Framework for the Internet of Things." In Proc. of International Conference on Computers, Privacy and Data Protection 2016 (CDPD 2016), volume 36 of LGTS. Springer, 2016.

[22] N. Doty, M. Gupta, "Privacy design patterns and anti-patterns." In: Workshop "A Turn for the Worse: Trustbusters for User Interfaces Workshop" at SOUPS 2013, 2013.

[23] J. Laprie, "Dependable computing and fault-tolerance." in Digest of Papers FTCS-15, 1985.

[24] A. Geraci, F. Katki, L. McMonegal, B. Meyer, J. Lane, P. Wilson, J. Radatz, M. Yee, H. Porteous and F. Springsteel, "IEEE standard computer dictionary: Compilation of IEEE standard computer glossaries." 1991.

[25] PP. Park, P. Di Marco, C. Fischione and K. Johansson, "Modeling and optimization of the IEEE 802.15. 4 protocol for reliable and timely communications." Parallel and Distributed Systems, vol. 24, 2013.

[26] J. Chen, J. Chen, F. Xu, M. Yin and W. Zhang, "When Software Defined Networks Meet Fault Tolerance: A Survey." Springer International Publishing, pp. 351–368, 2015.

[27] G. Aful, "Definition: Interoperability." Available: http://interoperability-definition.info/en/, 2018.

[28] G. Hatzivasilis, I. Askoxylakis, G. Alexandris, D. Anicic, A. Bröring, V. Kulkarni, K. Fysarakis and G. Spanoudakis, "The Interoperability of Things: Interoperable solutions as an enabler for IoT and Web 3.0." IEEE International Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks 2018 (IEEE CAMAD 2018), Barcelona, Spain, Sept. 17–19, 2018

[29] J. Manyika *et al.*, 2015. Unlocking the potential of the Internet of Things. McKinsey Global Institute Report, McKinsey & Company, June 2015, pp. 1–4, 2015.

[30] A. Al-Fuqaha *et al.*, 2015. Internet of Things: a survey on enabling technologies, protocols, and applications, IEEE Communication Surveys & Tutorials, IEEE, vol. 17, issue 4, pp. 2347–2376.

[31] E. Palavras, K. Fysarakis, I. Papaefstathiou and I. Askoxylakis, "SeMIBIoT: Secure Multi-Protocol Integration Bridge for the IoT." 2018 IEEE International Conference on Communications (ICC), pp. 1–7, 2018.

[32] http://www.opengeospatial.org/

[33] https://www.w3.org/2001/sw/

[34] "The SSN ontology of the W3C semantic sensor network incubator group." Web semantics: science, services and agents on the World Wide Web 17 (2012): 25–32.

[35] M. Bermudez-Edo *et al.*, "IoT-Lite: a lightweight semantic model for the Internet of Things." Intl. IEEE Conferences, Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016.

[36] J. Kiljander *et al.*, Semantic interoperability architecture for pervasive computing and Internet of Things. IEEE Access, IEEE, vol. 2, pp. 856–873, 2014.

[37] A. Bröring *et al.*, Enabling IoT ecosystems through platform interoperability. IEEE Software, IEEE, vol. 34, issue 1, pp. 54–61, 2017.

[38] M. Bartoletti *et al.*, "Semantics-based design for secure web services." IEEE Trans. on Software Engineering, 2008.

[39] M. Deubler *et al.*, "Sound development of secure service-based systems." In Proc. of the 2nd Int. Conf. on Service oriented computing. ACM, 2004.

[40] G. Geor *et al.*, "Verification and trade-off analysis of security properties in UML system models." IEEE Trans. on Software Engineering, 36(3): 338–356, 2010.

[41] J. Dong *et al.*, "Automated verification of security pattern compositions." Inf. Softw. Technol., vol. 52, no. 3, 2010.
[42] I. Siveroni, A. Zisman and G. Spanoudakis, "A UML-Based Static Verification Framework for Security, Requirements." Engineering Journal, 15(1): 95–118, 2010.
[43] S. Rossi, "Model checking adaptive multilevel service compositions." International Workshop of Formal Aspects of Component Software, 2010.
[44] A. R. Souza *et al.*, "Incorporating Security Requirements into Service Composition: From Modelling to Execution." In ICSOC-ServiceWave'09, 2009.
[45] L. Pino, K. Mahbub and G. Spanoudakis, "Designing Secure Service Workflows in BPEL." International Conference on Service-Oriented Computing, 2014.
[46] C. Alexander, S. Ishikawa and M. Silverstein, "A Pattern Language: Towns, Buildings, Construction." Oxford University Press, Oxford, 1977.
[47] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software." Addison-Wesley, Boston, 1994.
[48] M. Schumacher *et al.*, "Security Patterns – Integrating Security and Systems Engineering." Wiley, West Sussex, 2006.
[49] M. Hafiz, "A collection of privacy design patterns." In: Proceedings of the 2006 Conference on Pattern Languages of Programs, PLoP 2006, pp. 7:1–7:13. ACM, New York, 2006.
[50] L. Pino *et al.*, "Discovering Secure Service Compositions." 4th International Conference on Cloud Computing and Services Sciences (CLOSER 2014), Barcelona, Spain, 2014.
[51] L. Pino *et al.*, "Pattern Based Design and Verification of Secure Service Compositions." IEEE Trans. on Services Computing, 2017.
[52] A. Maña *et al.*, "Extensions to Pattern Formats for Cyber Physical Systems." Proceedings of the 31st Conference on Pattern Languages of Programs (PLoP'14), 2014.
[53] N. Petroulakis *et al.*, "Patterns for the design of secure and dependable software defined networks." Computer Networks 109 (2016): 39–49, 2016.
[54] N. Petroulakis *et al.*, "Fault Tolerance Using an SDN Pattern Framework." 2017 IEEE Global Communications Conference (GLOBECOM), 2017.
[55] T. Lorünser *et al.*, "Towards a new paradigm for privacy and security in cloud services." In: CSP Forum 2015. CCIS, vol. 530, pp. 14–25. Springer, Heidelberg, 2015.

[56] T. Länger, H. C. Pöhls and S. Ghernaouti, "Selected Cloud Security Patterns to Improve End User Security and Privacy in Public Clouds." In Proc. of Privacy Technologies and Policy – 4th Annual Privacy Forum (APF 2016), Springer, 2016.
[57] Business Rules Management System (BRMS), https://www.drools.org
[58] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem." Artif. Intell., vol. 19, no. 1, pp. 17–37, Sep. 1982.
[59] J. Seeger, R. A. Deshmukh, V. Sarafov and A. Bröring, "Dynamic IoT Choreographies – Managing Discovery," Distribution, Failure and Reconfiguration. IEEE Pervasive Computing, 18(1), pp. 19–27, 2019.
[60] A. S. Thuluva, A. Bröring, G. P. Medagoda Hettige Don, D. Anicic and J. Seeger, "Recipes for IoT Applications." Proceedings of the 7th International Conference on the Internet of Things (IoT 2017), 22–25. October 2017, Linz, Austria. ACM, 2017.
[61] A. Bröring, J. Seeger, M. Papoutsakis, K. Fysarakis and A. Caracalli, "Networking-Aware IoT Application Development." Sensors 2020, 20(3).
[62] J. Seeger, A. Bröring, M.-O. Pahl and E. Sakic, "Rule-Based Translation of Application-Level QoS Constraints into SDN Configurations for the IoT." EuCNC 2019, 18–21. June, Valencia, Spain. IEEE, 2019.