IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 20, 2021, accepted October 30, 2021, date of publication November 8, 2021,
date of current version November 23, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3125979*

# IoTsecM: A UML/SysML Extension for Internet of Things Security Modeling

**PONCIANO JORGE ESCAMILLA-AMBROSIO[1], (Senior Member, IEEE), DAVID ALEJANDRO ROBLES-RAMÍREZ[1], THEO TRYFONAS[2], ABRAHAM RODRÍGUEZ-MOTA[1], GINA GALLEGOS-GARCÍA[1], AND MOISÉS SALINAS-ROSALES[1]**

[1]Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico City 07738, Mexico
[2]Faculty of Engineering, University of Bristol, Bristol BS8 1TR, U.K.

Corresponding author: Ponciano Jorge Escamilla-Ambrosio (pescamilla@cic.ipn.mx)

This work was supported in part by the Consejo Nacional de Ciencia y Tecnología (CONACY), and in part by the Instituto Politécnico Nacional under Grant SIP 1999 and Grant 20210039. The work of Theo Tryfonas was supported by the Department for Business, Energy and Industrial Strategy, U.K.

**ABSTRACT** In this paper, an approach referred to as IoTsecM is proposed. This proposal is a UML/SysML extension for security requirements modeling within the analysis stage in a waterfall development life cycle in a Model-Based Systems Engineering Approach. IoTsecM allows the security requirements representation in two very well-known modeling languages, UML and SysML. With the utilization of this extension, IoT developers can consider the security requirements from the analysis stage in the design process of IoT systems. IoTsecM allows IoT systems to be designed considering possible threats and the corresponding security requirements analysis. The applicability of IoTsecM is demonstrated through applying it to analyze and represent the security requirements in an IoT real-life system in the context of collaborative autonomous vehicles in smart cities. In this use case, IoTsecM was able to represent the security requirements identified within the system architecture elements, in which all countermeasures identified were depicted using the proposed IoTsecM profile.

**INDEX TERMS** Cybersecurity, UML, SysML, Internet of Things, smart cities, autonomous vehicles.

## I. INTRODUCTION

The Internet of Things (IoT) represents a radical transformation of the existing Internet into an interconnected network of ''Smart Objects,'' generically referred to as ''Things.'' IoT systems not only collect data from the environment (they have sensing capabilities) and interact with the physical world (they can perform actuation, command, and control over other things), but also use the Internet to provide services for information transfer, processing, analytics, storage, and applications [1]. The main postulate of the IoT is that everything can be connected to the Internet anytime, anywhere, and using any network [2]. Hence, more objects such as smart cameras, wearables, environmental sensors, home appliances, and vehicles, are connected to the Internet every day. These connected things generate massive amounts of data among an increased number of IoT users, services, and applications across different domains. The collection, integration, processing, and analytics of these data enable the realization of smart cities, infrastructures, and services for enhancing the quality of life. A prediction from Gartner reported that by 2020 there would be 20.4 billion connected things [3]. These connected things cover a broad range of applications, for example, smart cities, smart grid, smart farming, smart health, among many more [1], [4].

In terms of security, identifying the vulnerabilities in an IoT system is strongly related to the system dimension. In other words, the attack surface grows when more elements are added to an IoT system, as these can be a point of access for an attack or an intrusion. Hence, identifying security requirements and depicting security controls in IoT systems impose new challenges that were not present in the traditional Internet. Furthermore, IoT systems' security requirements are frequently reviewed as an after-thought, even when the information handled by these systems is very sensitive in most cases. Therefore, understanding the associated security threats of IoT systems and identifying their potential solutions is imperative. In [4], a review of the main security

The associate editor coordinating the review of this manuscript and approving it for publication was Moussa Ayyash.

threats in IoT and cyber-attacks performed on IoT applications was presented. This empirical review identified several security requirements for IoT, such as resilience to attacks, data authentication, access control, client privacy, user identification, identity management, secure data communication, availability, secure network access, secure content, secure execution environment and tamper resistance.

However, IoT security is not a one-size-fits-all problem, and the solutions deployed to solve this problem tend to be quick fixes that do not consider all aspects needed. Hence, it should be recognized that IoT security is both multi-faceted and dependent on the effort to standardize IoT security tools.

In this context, an approach referred to as Internet of Things Security Modeling (IoTsecM) is proposed in this work. IoTsecM is a UML/SysML extension, which applies UML stereotypes mechanisms, UML/SysML diagrams and UMLsec stereotypes. IoTsecM aims to model security requirements in IoT systems to guide developers along the life cycle of the IoT systems design within the analysis stage in a Model-Based Systems Engineering (MBSE) approach. This work proposes a graphic representation of security modules, a nomenclature that encapsulates the IoT security requirements and UML diagrams extensions.

The first version of this approach was presented in [5], which was mostly a work in progress. The present work describes in detail the components of the proposed UML/SysML extension. Furthermore, to validate the applicability of IoTsecM, it has been employed to model the security requirements within the Flourish project [6], whose objective is to find innovative solutions related to customers interaction, connectivity, data analytics, and safe design for collaborative autonomous vehicles (CAVs) in a smart city domain. There are many threats in the Flourish environment, numerous cyber-attacks may target the Flourish assets, and they are also exposed to many motivated and not motivated attackers. Since the CAVs will be moving along the city, they can be easily reached as also its communication flow. The use of the IoTsecM profile makes it possible to identify the overall system security requirements, which then helps to place and depict the security mechanisms within the system architecture.

The remainder of the paper is organized as follows. In Section 2, related works about security modeling in IoT systems are reviewed. In Section 3, the IoTsecM approach is presented. Section 4 describes the application of IoTsecM to cyber security modeling in a CAV system within the Flourish project. Finally, Section 5 draws conclusions from the presented work.

## II. RELATED WORKS

Several approaches consider IoT security according to different viewpoints. From the security requirements point of view, the need for confidentiality, integrity, and authentication mainly depends on the security goals for each application at hand. Hence, there are many security requirements within the IoT applications ecosystem; for example, in [7] a table of security requirements from the infrastructure point of view was presented (see Table 1). In that proposal, the IoT environment was split into three categories: System Dependability, Communication Stack, and User and Service Privacy. For each of these categories, the authors found some requirements related to Confidentiality, Integrity, and Availability. Then they proposed some security components to target the security goals: AuthN (authentication module), AuthZ (authorization module), IM (identity management), KEM (key exchange management), and TRA (trust and reputation authority). However, the IoT involves more security requirements; for instance, tamper protection is needed for the physical layer in many scenarios.

**TABLE 1. Security requirements from the security infrastructure point of view.**

| System Dependability | Communication Stack | | User and Service Privacy |
|---|---|---|---|
| | Network Layer | Service Layer | |
| Service availability | Network-level anonymization | Service access control/Authorization | User privacy protection when using infrastructure |
| Infrastructure availability | Confidentiality | Service authentication | User privacy protection when using services |
| Infrastructure integrity | | Service reputation metering | Privacy protection of service towards user |
| Infrastructure trust | | Service trust | |
| Non-repudiation | | | |
| Accountability | | | |

In [8], some high-level security requirements were derived: resilience to attacks, data authentication, access control, client privacy, user identification, secure storage, identity management, secure data communication, availability, secure network access, secure content, secure execution environment, and tamper resistance. In this approach, more security concerns for IoT systems were considered, for instance, the different layer's requirements, according to their respective IoT architecture.

In [9], [10], security requirements for IoT systems have been obtained considering the Industrial Internet Reference Architecture (IIRA) point of view. The authors envision four viewpoints in their work: business viewpoint, usage viewpoint, functional viewpoint, and implementation viewpoint. From the business viewpoint, they are focused on the return of investment for security; in that sense, operations must be protected against the risk of damage. This damage may include interruption or stoppage of operations, destruction of systems, and leaking sensitive business and personal data resulting in loss of intellectual property, harm to the business reputation, and loss of customers. For the usage viewpoint, they propose security monitoring, security auditing, security policy management, and cryptography. For the functional

viewpoint, they provide six interacting building blocks organized into three layers. The top layer comprises the four core security functions: endpoint protection, communications and connectivity protection, security monitoring and analysis, and security configuration and management. These four functions are supported by a data protection layer and a system-wide security model and policy layer. For the implementation viewpoint, the authors outline common security issues: end-to-end security from the edge to the cloud, hardening of endpoint devices, protecting communications, confidentiality and privacy of data collected, managing, and controlling policies and updates, and using analytics and remote access to manage and monitor the entire security process.

From the modeling point of view, existing system modeling tools have been adapted to depict IoT systems through extensions of UML and/or SysML. An IoT-specific domain modeling language based on UML is proposed by Eterovic *et al*. [11]. The proposal represents things with labeled rectangles. These things contain one or more items representing sensors, actuators, or other components, and the communication between items is over ''provided'' or ''required'' interfaces. A circle and semicircle notation are adopted to represent the items' interfaces in a friendlier way rather than the traditional stereotypes in UML.

The authors in [12] present the approach referred to as UML4IoT, a UML profile focused on modeling cyber-physical components and their effective integration into IoT systems in the manufacturing application domain. The proposed approach is used to automate the transformation of cyber-physical components to an Industrial Automation Thing, i.e., a component with an IoT wrapper ready to be integrated into the modern IoT manufacturing environment. For the usability validation of their approach, the authors used a prototype implementation of the myLiqueur production laboratory system [13], which is, roughly, an IoT system that allows users to custom their liqueur by using a smartphone app remotely.

The UMLsec approach, presented in [14], is a UML extension based on a formal semantics to model computer system security properties. UMLsec aims to support secure system development through five goals in an already modeled system (using UML). UMLsec should be able to evaluate the system for security-related vulnerabilities in the design automatically. The authors argue that UMLsec defines precise semantics for security; it allows constructing a single formal description for the system model, including information from all diagrams and all abstraction layers.

Moreover, some authors have provided evidence supporting the potential benefits of linking cyber security techniques to MBSE and SysML; in this sense, in [15], an approach for extending SysML to be security-aware is discussed where, as stated by the authors, aims to provide a basis for discussion and development. The paper concludes by highlighting the advantage of using SysML over other modeling languages for industrial control systems (ICSs) modeling and the benefits of encouraging model-based systems engineers to consider security as one of the core concerns of system design.

In [16], an approach for the modeling, specification, and analysis of application-specific security requirements is presented. The proposal is based on a goal-oriented framework for generating and resolving obstacles to goal satisfaction focusing on security engineering at the application layer. From the threat trees point of view, used for modeling or documenting potential attacks in security-critical systems, they are built systematically through anti-goal refinement. Roughly speaking, it is achieved by introducing a formal epistemic specification that may support a formal derivation and analysis process.

Uzonov *et al.* [17] presented a pattern-driven security methodology (referred to as ASE) designed for distributed systems that also can consider peer-to-peer systems. The methodology is particularly focused on the design phase of these systems. It uses the principle of encapsulation by employing patterns to incorporate security features and threats modeling. The approach is illustrated in the development of a distributed system for file sharing and collaborative editing. The authors argue that ASE can address all or most of the core distributed systems security concerns, providing developers with detailed guidance on how and where to introduce relevant security features into a system's architecture during development.

In [18], Apvrille and Roudier presented SysML-Sec as a SysML environment that introduces diagrams for security matters and an associated methodology. They propose the stereotype ≪security requirement≫ which is used in the requirements diagram of SysML. The same authors in [19] further developed SysML-Sec to include attack graphs. The work proposed using the SysML's parametric diagram to depict attacks and their composition and represent the assets target of these attacks. The application of the proposed approach is illustrated on a PC and mobile malware examples. SysML-Sec seemed a useful extension because it can model security matters properly, but it lacks IoT semantics.

Some of the previous approaches consider a security layer but mostly as an after-thought: they do not consider a threat analysis and do not consider the IoT system security requirements and the wider attack surface of IoT systems.

Differing from previous works, we propose an approach referred to as IoTsecM. It is for security requirements modeling within the analysis stage in a waterfall development life cycle in an MBSE approach. It is achieved by allowing the security requirements representation in two very well-known modeling languages, UML and SysML.

## III. IoTsecM SECURITY MODELING

In this section, an extension approach to UML/SysML is introduced. This proposal is referred to as IoTsecM (IoT Security Modeling) since it aims to model the security requirements of IoT systems. Firstly, from a security analysis, fourteen security elements were identified [20], [21], which are abstract enough to depict the security concerns of

IoT systems from the analysis stage in an MBSE process. Hence, IoTsecM intends to allow developers to add security mechanisms in subsequent stages, such as design, even if the developers are not cybersecurity experts.

Once the security concerns were classified and depicted as elements, a nomenclature was proposed where each abstract security element was bounded to a nomenclature element.

A UML/SysML extension was chosen to deploy the proposed security nomenclature to provide an intuitive and graphic notation meant to decide where to introduce security countermeasures in an IoT system.

IoTsecM has two main contributions:

1) IoTsecM actors: they are introduced in section 3.1; they model the principal actors in an IoT environment (humans, actuators, sensors, tags, and IoT devices).

2) IoTsecM nomenclature: it comprises fourteen security elements. It is the IoTsecM core, introduced in section 3.2.

The IoTsecM profile design development starts once the IoTsecM metamodel is designed and the security nomenclature and actors are obtained, fitting it into a UML/SysML profile. In order to construct the associated metamodel, Eclipse Papyrus [22] was used, which is a UML tool that allows profile generation. There, the stereotypes, constraints, and tags were defined. In the following sections, the IoTsecM profile will be explained in detail. An overview of the IoTsecM profile nomenclature is displayed in Fig. 1, while an overview of the IoTsecM profile actors is shown in Fig. 2. The IoTsecM profile applies the UML and SysML metamodels to obtain the metaclasses' features from each one. However, it is necessary to clarify that, from SysML, the presented approach only extends the block and requirement metaclasses in the stereotypes proposed. For a broader explanation of IoTsecM, the reader is referred to [20].

### A. IoTsecM ACTORS

The IoT landscape comprises an enormous compendium of items able to interact in many ways. Therefore, there are many actors from many domains. Some approaches have tried to help in the actor's classification, such as the IoT-A proposal [7], specifically in their domain model, they identified users and devices.

The proposal of Actors does not rely on the security concerns directly. However, they are fundamental in the modeling stages since they abstract the main features of each IoT component, allowing the modeling process. IoTsecM contains five Actors regarded as:

- User: users can be humans or any digital device, application, service, or software agent that interacts indirectly or directly with the physical entity or the system.
- Sensor: any device that provides information about the physical entity.
- Actuator: any device that can modify the physical state of a physical entity.
- Tag: is typically attached to the physical entity and allows its identification.
- IoT device: is the hub and processing core element that gathers the sensor information and processes it. The IoT device handles the communication from the virtual entity to the system. It can send instructions to actuators.

The UML metaclass extension Actor and UML stereotypes are used to identify an actor, which defines the IoTsecM profile, as depicted in Fig. 2.

The IoTsecM stereotypes used to depict actors are: ≪IoTdevice≫, ≪User≫, ≪Actuator≫, ≪Sensor≫ and ≪Tag≫. They model the actors described earlier. A description of the IoTsecM Actors, including constraints and features, is given in the following paragraphs.

#### 1) IoTdevice

Extends the UML metaclass Device (from the deployment diagram) as it can be used to model the system architecture in terms of hardware. ≪IoTdevice≫ stereotype extends the Actor and Class metaclasses as well. This means that it can be applied to model an actor interacting with the system or a class in a UML class diagram. An IoT device has some predefined attributes that provide an abstract representation of its communication capabilities, such as:

- Bluetooth: this is a Boolean type value (True or False), but it allows the user to specify the value Undefined, providing developers with some flexibility when applying the profile. The TRUE value indicates that the Bluetooth communication is activated, while the FALSE value means that this capability is not supported or is not activated if it is supported.
- WiFi: this attribute is better defined as Boolean type, although it allows the Undefined value, as is the case for the Bluetooth attribute.
- MobileNetwork: this attribute is a Boolean type of value that defines whether the mobile network communication is activated or not.
- Zigbee: is a widespread communication protocol in wireless sensor networks (WSN); therefore, it is a predefined attribute of a Boolean type that represents whether the service is enabled or disabled. The Undefined value is also allowed.
- USBPort: many IoT devices have USB ports; hence, an attribute representing this condition has been defined as a Boolean type value.
- Microphone: if an IoT device is equipped with a microphone, it can be depicted with this attribute defined as a Boolean type.
- HDMI: it depicts a High Definition Multimedia Interface (HDMI) port, which is shown as a Boolean attribute.

Two operations are considered as predefined when the ≪IoTdevice≫ stereotype is used to depict the most common functionalities of this module:

- receiveSensorData(): indicates that the IoT device is able to gather sensor data.

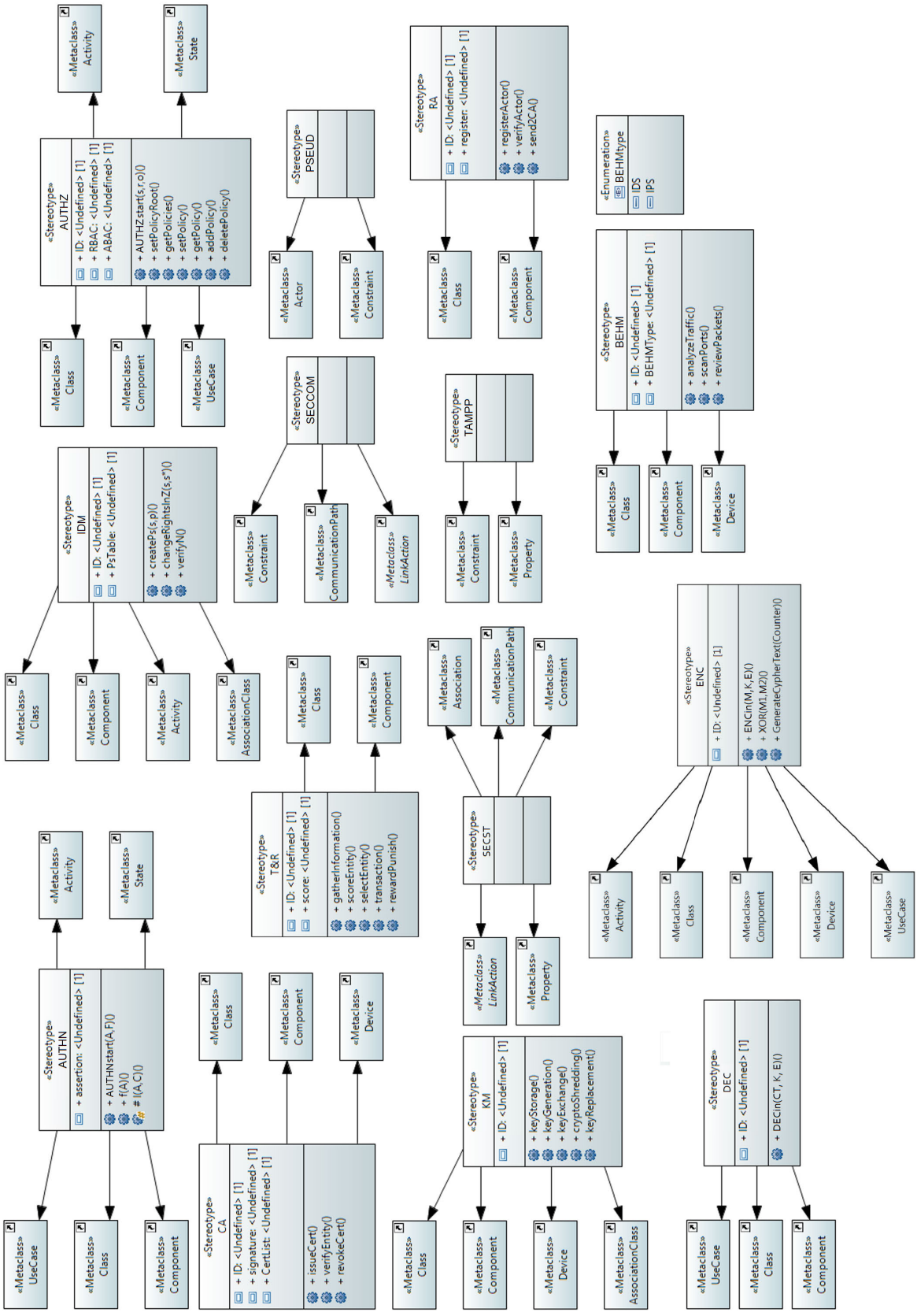


**FIGURE 1.** IoTsecM profile nomenclature overview.

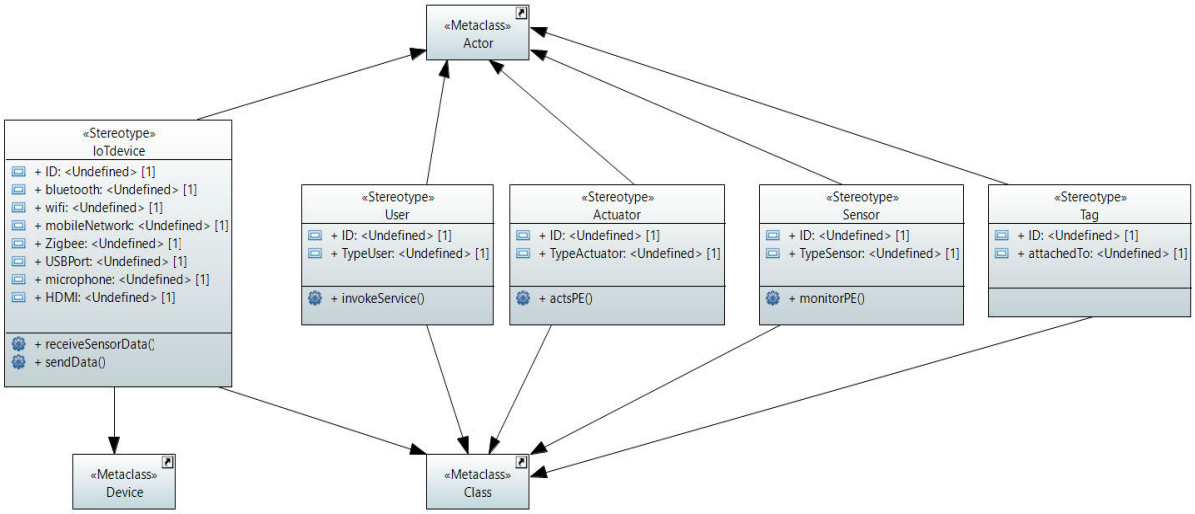


**FIGURE 2.** IoTsecM profile actors.

- sendData(): is a more general operation, representing whether the device is able to send data to the user or not. This operation can also represent that a device can send information to some external server or the cloud.

The ≪User≫ stereotype is applied to display the existence of a user, which, as mentioned earlier, may be a human, software, or any electronic device that invokes a resource from the physical entity. The ≪User≫ stereotype extends the actor and class metaclasses; thus, it can be applied as an actor in a use case diagram and over a class in a class diagram or in an object diagram when instances of it appear.

Two attributes are proposed for the ≪User≫ stereotype: ID and TypeUser. The ID attribute is unique and is recommendable to state it as Integer. Although in the stereotype definition, it is undefined to represent the case where the user is not linked to any technology or programming language. The TypeUser attribute is normally a string, and it defines the type of user who is invoking the resource such as human, software, device, etc. The operation offered by the ≪User≫ stereotype is invokeService() which invokes a service from the IoTdevice or another server.

#### 2) ACTUATOR

The ≪Actuator≫ stereotype refers to the entity which can modify the physical entity state. It has two attributes, the ID attribute, and the TypeActuator attribute. It admits the actsPE() operation, which is the abstract method defined to interact directly with the physical entity.

#### 3) SENSOR

The ≪Sensor≫ stereotype models the sensor actor, where there is a unique ID attribute for each ≪Sensor≫ instance. It defines a TypeSensor attribute where the specific sensor type is defined, for example, humidity, light, camera, etc. The only predefined operation is monitorPE(), which is the dedicated method to monitor some physical entity characteristic according to the type of sensor.

#### 4) TAG

The ≪Tag≫ stereotype models the tag actor; it includes a unique ID attribute for each tag. It also defines the attachedTo attribute, which indicates to which physical entity it is attached.

### *B. IoTsecM NOMENCLATURE*

The IoTsecM profile aims to address the security concerns of IoT systems. Therefore, the core of the proposal is a set of security elements, which are described in this section. These security elements encapsulate the security requirements. Besides, each element is part of a nomenclature, which allows us to depict them in a UML/SysML profile, making them easy to memorize. This profile helps to reduce box sizes in a diagram and allows a more agile design.

Fourteen elements have been identified and incorporated into the representation by applying a UML/SysML extension mechanism. A summary of these elements is presented in Table 2. The following subsections describe each IoTsecM nomenclature element.

#### 1) AUTHENTICATION: AUTHN

Authentication is the security service for ensuring that the identity of an entity (a user or service) is valid. Hence, it is an essential element of a typical security model. Authentication can be achieved by a mechanism that develops the authentication process of verifying the identity of an entity to prove if someone or something is, indeed, who or what it claims to be; ''authentication is the binding of identity to an entity'' [23].

In IoT systems, the authentication service is not just implemented in one layer of the system; it can be spread alongside the multiple layers of the whole architecture. For instance, the

**TABLE 2.** IoTsecM profile nomenclature.

| Element | Name | Extension mechanism | Base meta-class(es) |
|---|---|---|---|
| AUTHN | Authentication | Stereotype | Class, use case, component, block, activity and state |
| AUTHZ | Authorization | Stereotype | Class, activity, component, block, state and use case |
| ENC | Encryption | Stereotype | Use case, component, block, class |
| DEC | Decryption | Stereotype | Use case, class and component |
| SECST | Secure Storage | Stereotype | Link, property, association, communication path and constraint |
| SCOM | Secure Communication | Stereotype | Constraint, communication path and link |
| KM | Key Management | Stereotype | Class, component, block, device and association class |
| T&R | Trust and Reputation | Stereotype | Class, block and component |
| IDM | Identity Management | Stereotype | Class, block, component, activity and association class |
| PSEUD | Pseudonym | Stereotype | Actor and constraint |
| CA | Certification Authority | Stereotype | Class, block, component and device |
| RA | Registration Authority | Stereotype | Class, block and component |
| TAMPP | Tamper Protection | Stereotype | Constraint and property |
| BEHM | Behavior Monitor | Stereotype | Class, block, component and device |

perception layer must authenticate sensors, tags, and actuators; or the service layer must validate the IoT devices' identities. Therefore, in IoT systems, the authentication service becomes more complex; hence, it needs to be part of an entire security infrastructure.

The authentication process considers three main aspects [24]:

- What the entity knows: this approach is also known as knowledge-based, and it refers to private information supplied by the subject, for instance, passwords or secret information.
- What the entity possesses: this is also known as possession-based, and it could be a badge or a card.
- What the entity is: this is biometric-based, including, for example, fingerprints or retinal characteristics.

The authentication process refers to obtaining related information from an entity, analyzing these data, and determining if it is associated with that entity. This means that the processing unit must store some information about the entity. It also suggests that mechanisms for data management are required.

The element AUTHN represents an authentication process in a model item guaranteeing the authentication of an actor applying an authentication mechanism at a particular time. The AUTHN stereotype depicts an authentication mechanism regarding the abstraction level, which allows to model this security requirement. Certainly, this helps IoT designers to represent this security requirement from the design stage.

For the IoTsecM actors, the AUTHN element is applied as a security requirement and is expressed writing AUTHN over

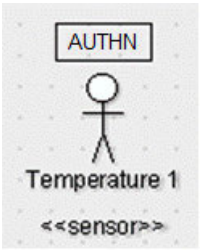


**FIGURE 3.** ≪AUTHN≫ stereotype depicted as a requirement over the actor's head.

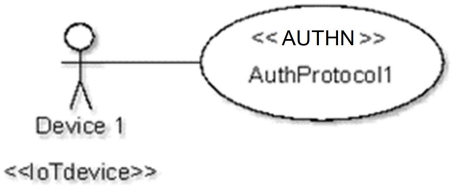


**FIGURE 4.** ≪AUTHN≫ stereotype represented as a use case.

the actor's head; this is an extension of the UML notation, since UML, SysML or SysMLsec do not depict security requirements in the use case diagram. This element will allow designers to depict the authenticated actors properly in a visual way and during an early stage of the system development. In Fig. 3, an authenticated actor is depicted, where AUTHN is written within a text box since it can be implemented in all UML/SysML tools.

If the authentication process is considered as a use case, then the ≪AUTHN≫ stereotype is annotated, as shown in Fig. 4. Here an entity is related to the use case, meaning that the entity stores an authentication process or protocol. For example, in Fig. 4 the ≪IoTdevice≫, named Device 1, authenticates other entities, e.g., sensors, actuators, or tags. At this level of abstraction, it could be difficult to determine which hardware or infrastructure element will address the authentication process. Hence, it can be observed that this representation fills the gap between the non-functional security requirements representation not addressed in UML or SysML approaches and the data or representation required by some automation tools or well-defined authentication protocols.

In brief, in a static diagram, the AUTHN stereotype helps better model a use case, class, software component, block (SysML), and object. Whereas, in a dynamic diagram the AUTHN element depicts an activity or a state. In Fig. 5, the ≪AUTHN≫ stereotype and the corresponding extended metaclasses are shown; note that the UML notation is preserved.

The AUTHN element has the $A$ and $f$ parameters as entry data. Therefore, the AUTHN element by itself or using another element of the nomenclature will determine $C$ to obtain $l$ [23]. The parameter $A$ is the set of specific information used by the entities to prove their identities; $C$ is the set of complementary information that is used by the

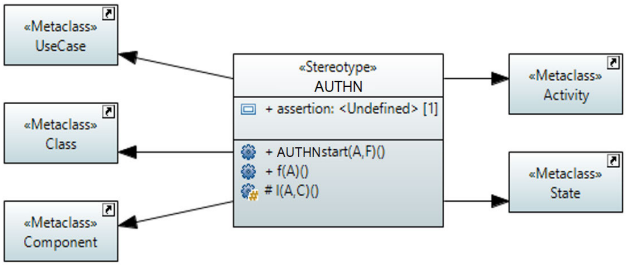


**FIGURE 5.** IoTsecM ≪AUTHN≫ stereotype and metaclasses extended.

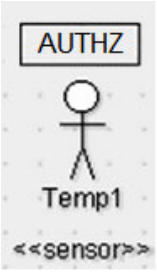


**FIGURE 6.** AUTHZ element for authorized actors.

system to validate authentication information; $F$ is the set of complementation functions that generate the complementary information from the authentication information; $L$ is the set of authentication functions used to verify identity, and $S$ is the set of selection functions that enable an entity to create or alter the authentication and complementary information.

With $l \in L$ and $f \in F$, this is:

$$AUTHN\ (A, f) = l(A, C) = \{True, False\} \quad (1)$$

$$f(A) = C \quad (2)$$

$$l(A, C) = \{True, False\} \quad (3)$$

The ≪AUTHN≫ stereotype can be a class; hence, it has instances (objects) which can be applied in a sequence diagram, object diagram, and state diagram. The AUTHN element also can be modeled as a software component using UML notation. In Fig. 5, the metamodel for the ≪AUTHN≫ stereotype is shown. The metaclasses extended are UseCase, Class, Component, Activity, and State. The ≪AUTHN≫ stereotype contains the three main operations described before and the assertion attribute, which results from the authentication process, which is recommendable to be declared as a Boolean type value.

The function of the AUTHN element is to provide designers an abstract module that helps them further implement of either a well-known or a novel authentication approach [25]. This implementation can also be modeled using IoTsecM state machine diagrams, where the life of an instance of the ≪AUTHN≫ stereotype is described step by step.

### 2) AUTHORIZATION: AUTHZ

Once an actor is identified and authenticated, we must determine the rights it is granted (read, write, delete, execute). Therefore, an authorization element is fundamental for the IoTsecM extension. The AUTHZ element refers to access control decisions based on access control policies.

AUTHZ relates to the capacity of authorizing or refusing a user or entity to access a resource, with some specific actions permitted according to the user identity. From an abstract point of view, its basic functional principle can be modeled like [21]:

$$AUTHZ\ (s, r, o) \rightarrow \{true, false\} \quad (4)$$

with $s \in S$, $r \in R$, and $o \in O$; $S$ is the set of subjects performing the access; $R$ is the set of resources to be accessed; $O$ is the set of operations to be performed on the resource.

The functionality of the AUTHZ element is based on the AuthZ component described in [7], here it is defined as:

$$\text{Boolean} : \text{AUTHZ.authorize (Assertion, Resource,} \times \text{ActionType}) \quad (5)$$

The result of the AUTHZ element decision is TRUE or FALSE (permit or deny). Assertion depicts the details of the actor who accesses the information, which typically are: ID, certificate, Security Assertion Markup Language (SAML) assertion, Kerberos assertion, etc. Resource indicates the asset to be accessed, for instance, services or data. ActionType depicts the action to be performed over the associated resource, for example, read, write, or execute.

In [26], a nomenclature for access control systems is defined, where the authorize functionality is called Policy Decision Point (PDP), the Policy Enforcement Point (PEP), and the Policy Administration Point (PAP). All these functionalities are encapsulated within the AUTHZ element since these are the basic access control components commonly utilized, providing the system model with an abstraction of the access control features.

Currently, two main access control approaches could be suitable for IoT systems: Role-based access control (RBAC) and Attribute-based access control (ABAC). IoTsecM provides an extension to depict whether an actor is authorized or not; it is achieved by writing AUTHZ over the actor's head, indicating that the current actor is or needs to be authorized. In Fig. 6, an example is shown where a sensor is annotated with AUTHZ over its head. This is the first step in the design stage to map an authorization requirement within the IoTsecM notation.

The functional principle described before is mostly related to Access Control Lists (ACLs) and its correct implementation; therefore, the AUTHZ element involves an authentication mechanism. The AUTHZ element must be able to map the certificates authenticated from AUTHN to some specific policies. An ≪AUTHZ≫ stereotype instance can be called from an ≪IDM≫ stereotype instance if it is resolving a pseudonym (see Section 3.A.8). In Fig. 7, an IoT device, called Device 1, is related to the use case AUTHZ, which depicts that this actor develops an authorization mechanism.

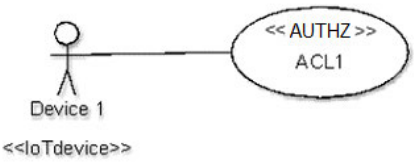


**FIGURE 7.** ≪AUTHZ≫ stereotype applied in a use case.

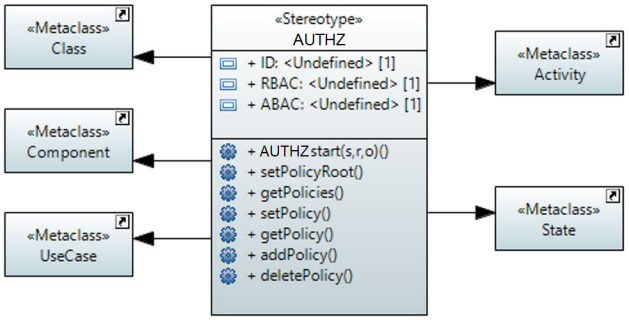


**FIGURE 8.** ≪AUTHZ≫ stereotype definition.

The AUTHZ element models the three main functionalities of an authorization infrastructure. Therefore, the main operations of the PAP should be applied in the AUTHZ element. The IoTsecM profile will allow the design of these abstract operations within the ≪AUTHZ≫ stereotype, as it is shown in Fig. 8, where the ≪AUTHZ≫ stereotype definition is shown and the metaclasses extended are Class, Component, UseCase, Activity, and State. The main functionalities of a PAP are predefined through the operations: *AUTHZstart*$(s, r, o)$, *setPolicyRoot*(), *getPolicies*(), *setPolicy*(), *getPolicy*(), *addPolicy*() and *deletePolicy* The ≪AUTHZ≫ stereotype includes three properties, two of them are represented by Boolean values, which are RBAC and ABAC that define the control access type property, plus the ID property.

### 3) ENC: ENCRYPTION AND DEC: DECRYPTION

Cryptography, defined as the study of mathematical techniques related to aspects of information security [27], provides cryptographic tools, also named primitives, for securing data, transactions, and personal privacy.

A cryptographic module offers the services of encryption, digital signature, and message authentication code (MAC), among other services, to achieve confidentiality, authentication, integrity, and non-repudiation security services [27]. Cryptographic primitive types fall into the following categories [28]:

- Encryption:
  - Symmetric
  - Asymmetric
- Hashing
- Digital signatures
  - Symmetric
  - Asymmetric
- Random number generation: The basis of most cryptography algorithms requires very large numbers originating from high entropy sources.

On the one hand, symmetric algorithms use the same shared cryptographic key to encrypt and decrypt data. On the other hand, asymmetric algorithms use a publicly known key for encryption but requires a different key, known only by the intended recipient, for decryption. From the cryptographic primitives, different algorithms should be selected, and then a cryptosystem is defined.

Representing an Encryption element using the ≪ENC≫ stereotype indicates the dynamic or static encryption of data, with the encryption algorithms to achieve the encryption of data contained on such element. This element can work with any other element which encrypts data. Therefore, the ≪ENC≫ stereotype is an abstraction of a cryptographic primitive module; it provides symmetric encryption, asymmetric encryption, counter modes, hashes, and digital signatures.

The ENC element receives the data, $M$ (or message), the encryption key, $K$, and the encryption algorithm, $E$, that will be applied; consequently, it returns the encrypted data (or ciphertext), $CT$, and the key used. For symmetric encryption, it returns the single encryption key, and in the case of asymmetric encryption, it returns the public encryption key used for the encryption process. The following function describes this process:

$$ENCout = ENCin(M, K, E) \tag{6}$$

where $M$ is the data (or message) to encrypt, $K$ is the encryption key, and $E$ is the algorithm required for the encryption process.

The XOR operation is used in blockchains and counter modes; besides, it helps other system functionalities. Therefore, the ENC element includes the XOR operation.

$$XORout = XORin(M1, M2) \tag{7}$$

where $M1$ and $M2$ are the data that will be XORed.

The counter modes make use of a *Counter*; in these, the plaintext data is not encrypted with the encryption algorithm and key; instead, each bit of plaintext is XORed with a stream of continuously produced ciphertext comprising encrypted counter values that continuously increment; this is:

$$ENCMout = Generate\_CypherText(Counter) \tag{8}$$

Although the encryption algorithm is not always performed by just one module, it is important to add a counter mode that is able to produce ciphertext.

In order to use block chaining modes, the ≪ENC≫ stereotype instance can be called as many times as the chain requires it. Hence, it can be easily added since the ENC element is a meta-class that can be applied and added following the system requirements. In this way, the cipher block chaining operation module (CBC) is included.

The ≪ENC≫ stereotype can be applied as a use case within the use case diagram to depict that a given actor


**FIGURE 9.** ≪ENC≫ use case example.

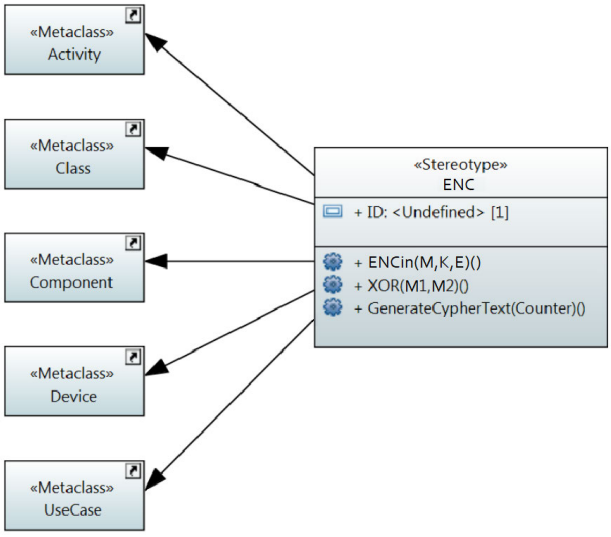


**FIGURE 10.** ≪ENC≫ stereotype definition.

has to encrypt data, or it could have an encryption module as well. In Fig. 9, the ≪ENC≫ stereotype is shown as a use case where an actor named CentralNode, which is an IoTdevice, must encrypt data. This means that it requires the minimum encryption capabilities described before according to the system security requirements and computing power.

The ≪ENC≫ stereotype extends the use case, class, component, device, activity, and block (SysML) metaclasses, see Fig. 10. This stereotype includes the abstract operations previously described, such as *ENCin*($M$, $K$, $E$). The second operation provided by the ≪ENC≫ stereotype is an XOR, depicted as *XOR*($M1$, $M2$). The last operation is *GenerateCypherText*(*Counter*) which corresponds to the counter mode explained earlier.

The main functionality of the DEC element is to decrypt the encrypted message; therefore, it must know the algorithm used in the encryption process to be able to decrypt the information. Hence, it knows the key used to encrypt the data, meaning communication between the DEC and KM elements is needed. Nevertheless, the architecture is not regarded in the metamodel proposed and the aim is to provide the abstract elements with abstract operations and attributes to be customized as each IoT system requires them.

The main functionality of the DEC element is the decryption of data, applying the function:

$$DECout = DECin(CT, K, E) \tag{9}$$

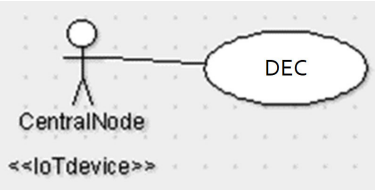


**FIGURE 11.** A DEC element as a use case.

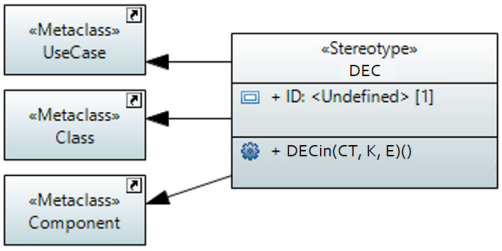


**FIGURE 12.** ≪DEC≫ stereotype definition.

where $CT$ is the Ciphertext, $K$ is the decryption key, and $E$ is the algorithm required for the decryption process.

Within the IoTsecM profile, the DEC element can be depicted as a use case; this means that an actor who includes this use case must have a decipher module with the algorithm used to decrypt the data. An example of this is shown in Fig. 11, where an actor named CentralNode has a DEC use case. This means that that this actor performs the decryption process of a given data.

The ≪DEC≫ stereotype is proposed to cover the decryption process in the analysis stage; therefore, it extends the UseCase, class, and component metaclasses from UML. On the other hand, it extends the block metaclass from SysML. The only property defined is the ID and the operation defined is the DECin operation, which encapsulates the ≪DEC≫ function described earlier. In Fig. 12, the ≪DEC≫ stereotype definition can be observed.

#### 4) SECST: SECURE STORAGE

As discussed before, within the IoT environment, there are resources on-device and in the network. In many scenarios, the stakeholders would like to protect that sensitive information or store it in a secure place.

Sensitive data can be protected by applying encryption at the field, directory, record, file system, or storage device level. Cryptographic algorithms use keys to encrypt and decrypt data blocks; nevertheless, key management imposes a hassle in IoT storage systems. The IoTsecM profile aims at covering the abstraction of a secure storage requirement. The IoTsecM profile proposes an extension to UML/SysML to depict the secure storage requirement. It is done by applying the stereotype extension mechanism entitled ≪SECST≫ which extends the constraint, link, association, communicationPath, and property metaclasses from UML, and the

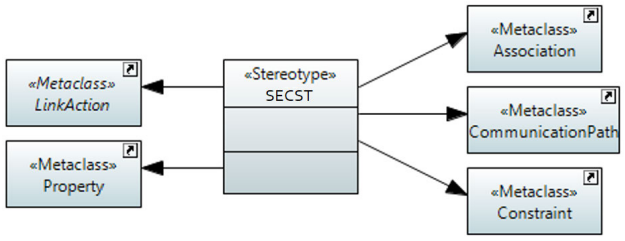


**FIGURE 13.** ≪SECST≫ stereotype definition.

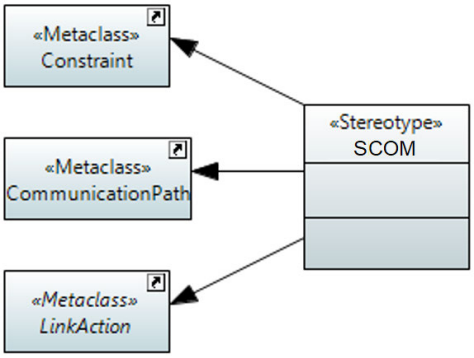


**FIGURE 14.** ≪SCOM≫ stereotype definition.

requirement metaclass from SysML. The ≪SECST≫ stereotype definition is shown in Fig. 13.

#### 5) SCOM: SECURE COMMUNICATION

In many cases, at the benchmark of projects, the security mechanisms to implement or consider are uncertain. At some point, developers can depict the first use cases and consequently the first security requirements in an abstract representation. When the secure communication requirement appears, then some parameters should be addressed. According to [7] the following parameters can be part of the secure communications enablement request:

- Target(s) identifier(s): entity identifiers with which the requesting node is trying to communicate in a secure way.
- Type of secure communications enablement: it may be Authenticated Key Exchange (AKE) protocol running between the communicating nodes. The request should establish whether the security property of Perfect Forward Secrecy (PFS) is required or not between the requesting node and the target node(s) with which it will securely communicate. PFS may mean that an especially robust AKE protocol will be triggered between the nodes.
- Type of authentication: the requesting node may wish to authenticate its peer(s) using an end-to-end scheme.
- Supported identification scheme(s): the infrastructure and both entities must support the identification scheme.

The proposed extension is based on the UMLsec ≪secure links≫ stereotype [14], where secure communication and information flow are considered. In the IoTsecM profile, the ≪SCOM≫ stereotype is presented, shown in Fig. 14, as a UML extension that declares the condition when the communications between two entities need to be secured. Besides, it is a constraint, and this means that the security requirement must be attended to in subsequent stages such as design. For instance, a certificate-based protocol may attend the {SCOM} constraint between two entities. Hence, the ≪SCOM≫ stereotype extends the link, communication path, and constraint metaclasses from UML and the requirement metaclass from SysML.

#### 6) KM: KEY MANAGEMENT

A key management system enables and assists IoT assets in the establishment of secure communication or context. It is an integrated approach to generate, store, and handle the keys within a cryptosystem. The KM element provides capabilities to the IoT assets to assist the low-resource nodes in their operations, and it depends on specific protocols and security mechanisms. This element should support the process of enhancing the secure communication between a user and a service by setting up a tunnel [7] between gateways, which is very useful for users and services running on low-resource devices. In the IoT environment, it is necessary to grant suitable key management mechanisms that allow two remote devices to exchange security credentials.

It is possible that a known key management system (KMS) may not apply in an IoT context, mainly because the user and service are in different networks. The common behavior of a KMS is to provide a trusted third party that provides the two entities with the corresponding keys. For example, if a user and an IoT device share the same secret key, the objective would be to provide this single secret key. This would follow the next simple protocol [23]:

- User to KM: {request for session key to IoTdevice} $k_{user}$
- KM to User: $\{k_{session}\}\, k_{user} \,||\, \{k_{session}\}\, k_{IoTdevice}$
- User to IoTdevice: $\{k_{session}\}\, k_{IoTdevice}$

The IoTdevice is now able to decrypt the message and uses Ksession to communicate with User. This is just an overview of the performance of a KMS; it has some vulnerabilities and, clearly, improvements which are explained in [23]. The KM element aims at describing and modeling the general behavior of a KMS; therefore, some abstract operations need to be considered, such as:

- Key storage: a set of keys is securely stored for subsequent use.
- Key generation: the KM element should be able to generate a proper key when requested.
- Key exchange: in some cases, identical keys need to be exchanged between two parties (symmetric key system); in other cases, the other party's public key may require to be shared (asymmetric keys).

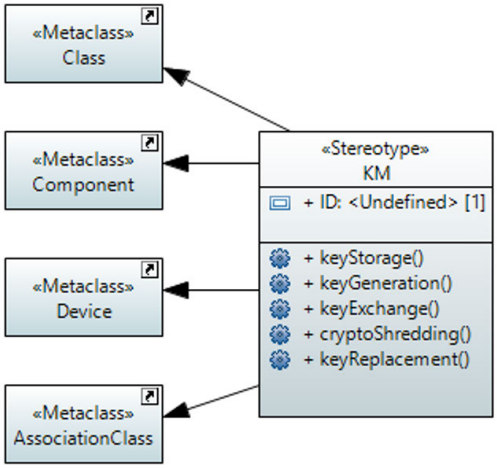


**FIGURE 15.** ≪KM≫ stereotype definition.

- Crypto-shredding: delete key data or revoke it.
- Key replacement: it should be able to replace a specific key.

The KM element is modeled by applying a stereotype named ≪KM≫, which is a UML extension mechanism. The operation described above is modeled by this stereotype and extends the following metaclasses: class, component, device, and block (from SysML). In Fig. 15, the ≪KM≫ stereotype definition is shown in a UML profile diagram. As can be seen, one attribute is defined, specifying an ID and the abstract instructions which correspond to the KMS functionalities introduced earlier. The defined operations are keyStorage, keyGeneration, keyExchange, cryptoShredding, and keyReplacement.

### 7) T&R: TRUST AND REPUTATION

According to [29], trust is a particular level of the subjective probability with which an agent will perform a particular action. Reputation is the expectation about an agent's behavior based on information about it or observations of its past behavior [30].

The most common functionalities of a trust and reputation (trust or reputation) model, as described in [7], are:

- Gathering information: collect behavioral information about the entities in the system.
- Scoring and ranking: once the entity information has been gathered, then it will be analyzed and scored.
- Entity selection: the scoring data help trusting entities decide which entities interact with each other and which are not reliable. In the IoT landscape, it would define sensor interactions.
- Transaction: once a sensor is selected, the transaction occurs between both entities giving a specific service.
- Reward and punishment: once a transaction is completed, the client entity may assess that transaction to reward or punish the entity which provided the service.

The evaluation of the reputation of an entity needs to consider the low-computational power of some IoT devices

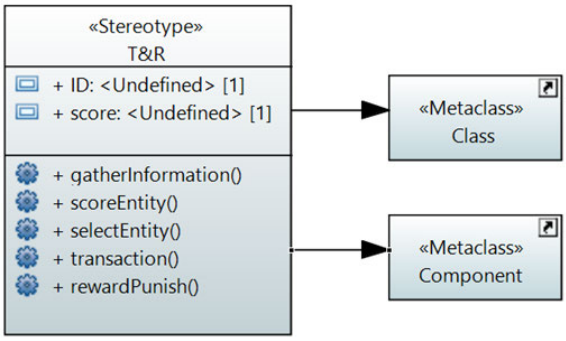


**FIGURE 16.** ≪T&R≫ stereotype definition.

such as sensors; hence such constraints need to be considered in this evaluation to be light, scalable, etc.

In the IoTsecM profile, the trust and reputation services are considered as an extension as well, where the extension mechanism applied is represented with the stereotype ≪T&R≫. The name should include the "or" word as well, although the name would not be very practical to write. Hence, ≪T&R≫ means trust and reputation, and trust or reputation. The ≪T&R≫ functionalities described before are depicted as operations within the stereotype definition, as shown in Fig. 16. The operations defined for this stereotype are gatherInformation, scoreEntity, selectEntity, transaction, and rewardPunish, corresponding to the general functionalities described above. The ≪T&R≫ stereotype includes two attributes, the entity ID and the score of the trust and reputation processes.

### 8) IDM: IDENTITY MANAGEMENT, AND PSEUD: PSEUDONYM

In IoTsecM, identity management (IDM) is based on the approach described in [7]. In many IoT scenarios, it is imperative to protect the identity of users, actors, etc. Therefore, information about the identity must be supplanted by applying a pseudonym. This element is related to the ≪PSEUD≫ stereotype since it is the entity that handles the pseudonyms and system identities. The IDM issues pseudonyms and accessory information to trusted subjects. This element protects user privacy and service privacy.

A pseudonym is a temporary identity for an imaginary subject that includes temporary credentials and access rights depending on the requesting subject; therefore, a pseudonym can request another pseudonym.

The pseudonym generation may be depicted as:

$$createPSEUD(s_1[s_1, s_2, s_3, \ldots, s_n], p \rightarrow s*) \qquad (10)$$

where $s_i$ is the subject or set of subjects, $s*$ is the requested pseudonym, $p$ is the set of specifications such as key, length, algorithm, access rights, etc.

The generated pseudonym preserves the subject access rights or, if it is requested, it may include fewer rights than the original subject, but it will never contain more rights. The expiration date possessed by the pseudonym is less than or

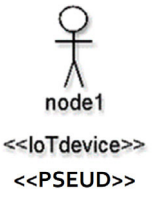


**FIGURE 17.** ≪Ps≫ stereotype example.

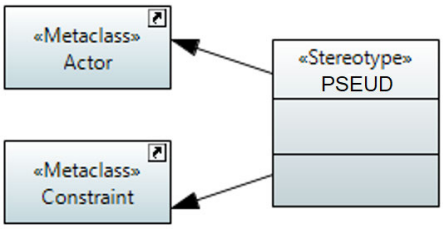


**FIGURE 18.** ≪PSEUD≫ stereotype example.

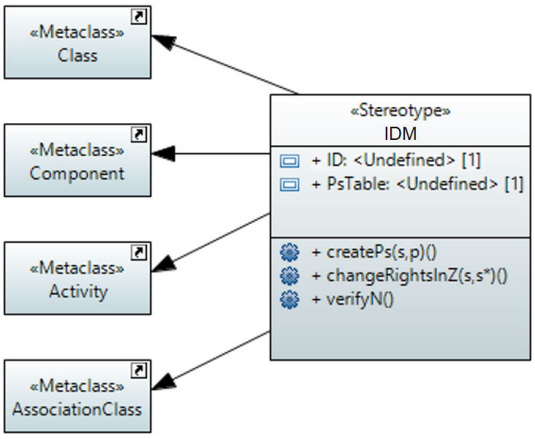


**FIGURE 19.** ≪IDM≫ stereotype definition.

equal to the subject's one. The request of a pseudonym should be only requested by a secure channel.

The IDM element is the only one that keeps track of the relationship between the pseudonyms and the subjects. Another functionality of the IDM is that, when creating a new pseudonym, it must update the access policies in the AUTHZ element; therefore, it also associates a new address to the ID.

Two stereotypes are defined to depict the pseudonyms' concerns, the ≪PSEUD≫ stereotype is an abbreviation of the pseudonym word, and it depicts the requirement of an actor of a pseudonym. For instance, if an ≪IoTdevice≫ entitled as "node1" requires a pseudonym, then ≪PSEUD≫ is depicted as shown in Fig. 17, this will help to identify those actors that protect their identity and, consequently, the actor's privacy.

The ≪PSEUD≫ stereotype extends the actor metaclass, the constraint metaclass, and the requirement metaclass from SysML. In Fig. 18 the ≪PSEUD≫ stereotype definition is shown.

The ≪IDM≫ stereotype is applied to depict the identity management mentioned earlier. It extends the class, component, activity, and association metaclasses. It defines three main operations: verification of the assertion in the corresponding ≪AUTHN≫ instance, the changeRightsInAUTHZ operation, which changes the root identity permissions, and assigns the previous ones to the pseudonym generated regarding the constraints mentioned before. The ≪IDM≫ stereotype has two attributes, see Fig. 19, the first one is the ID attribute, and the second one is a table (PsTable) which contains the links between the original root entities and the pseudonyms generated, it cannot be queried by any external entity, module, component, etc.

### 9) CA: CERTIFICATION AUTHORITY AND RA: REGISTRATION AUTHORITY

A Certification Authority (CA) is a trusted entity responsible for issuing and revoking digital certificates using a digital signature, where an asymmetric cryptographic algorithm is applied. The certificates include numeric IDs and passwords, besides making available the verification process to validate the provided certificates. The CA legitimates the relation between the actor identity and its public key to third entities that trust the CA certificates. Although a homogenized process to trust a CA does not exist, it is a fundamental concept for the correct performance, and then the entities who request a certificate from a CA must trust it.

A CA is generally applied in a Public Key Infrastructure (PKI), where before issuing a certificate, the CA must verify the identity of each actor requesting network access. In order to achieve it, the requesting actor delivers a Certificate Signing Request (CSR), which contains information about the organization requesting the certificate, a public key, and the digital signature created by the requestor's private key. Then the certificate is generated and signed by using the CA private key to allow all the network members to validate the authenticity of the certificate and the identity of its holder. Along with the entity ID, a digital certificate includes essential information related to the algorithm employed to create the signature, the digital signature of the CA, the purpose of the public key encryption, signature, and certificate validity interval. An example of a digital certificate according to the standard ITU X.509 is illustrated in Fig. 20.

The CA element is a legacy component, which provides certificates binding service from virtual entities to defined attributes.

In the IoTsecM profile, the CA element is regarded as an extension to UML/SysML. This modeling artifact is extended by a stereotype named ≪CA≫, see Fig. 21. The ≪CA≫ stereotype is thought to be applied as class, component, and device in UML; however, in SysML, it extends the block metaclass. The functionalities of the ≪CA≫ stereotype issue a certificate, verify the entity (this operation is with the registration authority), and revoke the certificate. In order to model these abstract functionalities three operations are proposed: issueCert, verifyEntity, and revokeCert, each one corresponding to the previous functionalities.

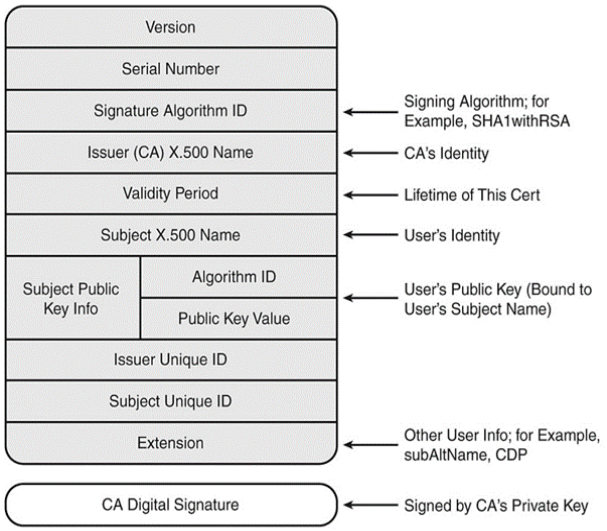


**FIGURE 20.** Certificate structure according to the ITU standard X.509.

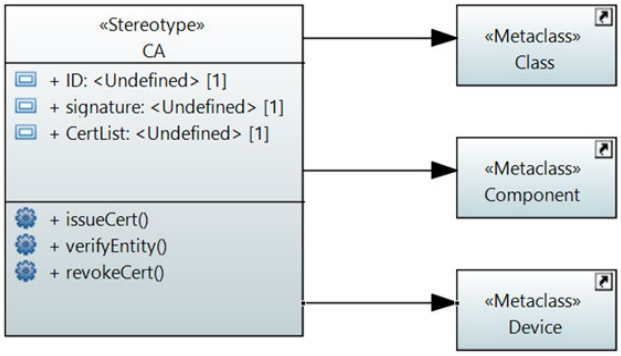


**FIGURE 21.** ≪CA≫ stereotype definition.

The ≪CA≫ stereotype can participate in verifying the identity of the message transmitter when certificates are supported. Based on the certificates, secure service-based communication can be established. Other elements such as AUTHZ, T&R, and AUTHN rely on this element to link their activities to the correct subjects.

The registration authority controls the certificate generation and is usually encapsulated in the CA; it realizes the certification petition and saves the corresponding data. Nevertheless, as IoTsecM aims to cover as many architectures and configurations as possible, it is regarded as another element named ≪RA≫, and it is a stereotype as well. The RA functionalities are:

- Register the user requests to obtain a certificate.
- Verify the user's data truthfulness.
- Send the request to a CA to be processed.

The ≪RA≫ stereotype extends, from UML, the class and component metaclasses, and from SysML, the block metaclass, see Fig. 22. It does not extend the device metaclass since it is usually encapsulated in the CA element. Nevertheless, to obtain a greater flexibility in the analysis

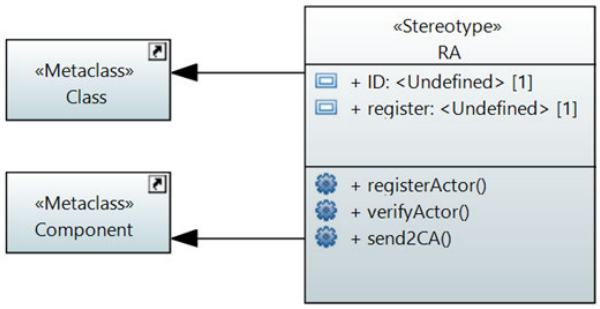


**FIGURE 22.** ≪RA≫ stereotype definition.

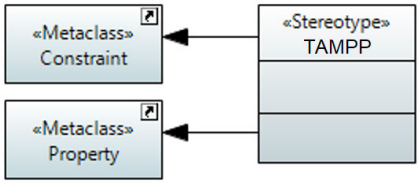


**FIGURE 23.** ≪TAMPP≫ stereotype definition.

and design stages, this stereotype extends the class and component metaclasses. The operations included by the ≪RA≫ stereotype are registerActor, verifyActor, and sen2CA, which model the functionalities introduced before. The attributes of the ≪RA≫ stereotype are an ID and a register, which is the record of the certificates to be issued.

### 10) TAMPP: TAMPER PROTECTION

The IoT environment involves many scenarios where IoT devices and sensors may be deployed in unreachable and exposed places; thus, an adversary might tamper with them and capture them to, for example, extract cryptographic information, modify programs, or replace them with malicious nodes. As it would be expensive and complicated to protect them with infrastructure or vigilance, the solution to the physical threats is to attempt to drive IoT devices procurements that include physical tamper protection [28]. Therefore, tamper-resistant packaging would assist the defense against the existing threats [31]. The tamper-resistant package would mitigate the physical attacks that threaten the confidentiality, integrity, and vulnerability information as well as the actor's privacy.

Another factor as important as the tamper-resistant one is tamper-detect. In [32], it is mentioned that systems should provide tamper-evident environments such that any physical or software tampering by an adversary is guaranteed to be detected.

In the IoTsecM profile, the tamper protection is regarded as a UML/SysML extension with the proposed stereotype named ≪TAMPP≫, which means tamper protection represents the security requirement of tampering resistance. Hence, this denotes that an entity containing the ≪TAMPP≫ stereotype must consider tamper protection since the analysis and design stage, to prevent it from being logically or physically altered.

The ≪TAMPP≫ stereotype, see Fig. 23, defines a security requirement of tamper protection for entities and system components (hardware and software). Thus, it extends the constraint and property metaclasses from UML and the requirement metaclass from SysML. Therefore the ≪TAMPP≫ stereotype can depict constraints in UML diagrams in order to indicate that a given entity requires to be tamper-proof, it is a property, and hence it can be displayed within a class property to indicate that such class or class property requires to be protected from tampering. The ≪TAMPP≫ stereotype is part of the IoTsecM profile and can be used in a secure development design process.

#### 11) BEHM: BEHAVIOR MONITOR

The IoT security requirements relate mainly to the first defense line, which is typically established by the ≪AUTHZ≫ and ≪AUTHN≫ stereotype instances. These security mechanisms provide security to some parts of the system. However, there is not a system without vulnerabilities due to inside or outside intruders that may exploit wireless communication protocols. Therefore, another defense line is needed where a security control can monitor the system behavior to detect the malicious behavior and then report it, and in some cases, react accordingly. In a passive system, when the behavior monitor detects a possible intrusion, it stores the information and sends an alert signal that is stored in a database. In a reactive system, the behavior monitor reacts to the suspicious activity, reprograming the firewall, if it is the case, or updating the policies within the ≪AUTHZ≫ stereotype to block the traffic which comes from the attacker [33]. The analogies to the classical security mechanisms are Intrusion Detection System (IDS) and Intrusion Protection System (IPS). However, applying traditional IDS techniques to IoT systems is difficult due to their characteristics, such as resource-constrained devices, specific protocol stacks, and standards. Finding nodes that comply with the computational resources to support IDSs and locate them in a place where the IDS is relevant is not an easy task when the system is being deployed. Therefore, they should be considered from the first design stage, helping to define an adequate architecture better and selecting devices that support the behavior monitoring. Within IoT systems, the behavior monitor can be in the border router, in one or more dedicated hosts, or in every physical object [34].

The IoTsecM profile proposes an abstract module that allows the behavior monitoring of some or specific system parts. The analysis of the network traffic, the port scanning, and the malformed packets are some of the analysis functionalities of this element. This approach addresses the behavior monitoring security requirement proposing a UML extension mechanism, which is a stereotype named ≪BEHM≫ that encapsulates all the semantics described earlier; moreover, it can be properly placed since the design stage.

The ≪BEHM≫ stereotype addresses the necessity of a behavior analyzer in the system extending the metaclasses: Component and Device Component from UML and the Block metaclass from SysML. The ≪BEHM≫ stereotype defines two properties: an ID and a BEHMType, which can be an IDS or IPS, see Fig. 24. There are three operations defined that correspond to the behavior of the ≪BEHM≫ stereotype, and these operations are analyzeTraffic, scanPorts, and reviwePackets.

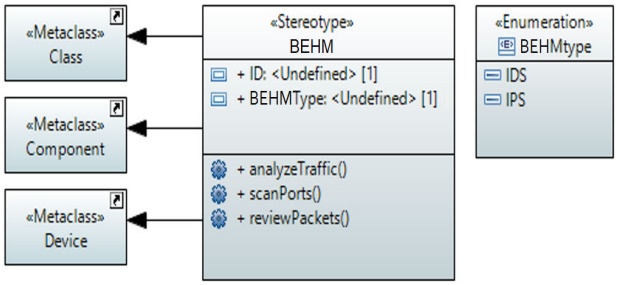


**FIGURE 24.** ≪BEHM≫ stereotype definition.

## IV. APPLICATION OF IoTSecM PROFILE TO COLLABORATIVE AUTONOMOUS VEHICLES

As described in the previous section, the IoTsecM profile addresses the design and modeling of IoT systems considering a security architecture, helping to depict the system security concerns. Once the possible attacks over a system are identified, developers can figure out how to provide protection or countermeasures against those attacks. Therefore, they would be able to find the right place for the proper countermeasure for an attack or threat.

There is no unique methodology for threat modeling that will help to mitigate all risks in a system. The main objective of threat modeling is to identify the system threats and vulnerabilities, which undoubtedly would be exploited by a motivated attacker if countermeasures are not there to prevent them. More about threat modeling can be found in [35]. Microsoft proposes another approach, referred to as Microsoft SDL [36], that uses multiple steps to determine the severity of threats.

In this sense, a use case applying IoTsecM for threat modeling in an autonomous vehicles system in the context of Smart Cities is presented in this section. The selected system is a real-life project named Flourish [6]. In this system, autonomous vehicles are considered, and the interaction between them and other assets, such as city infrastructure sensors and traffic lights. The main project's objective is to find innovative solutions for customer interaction, connectivity, data analytics, and safe design for collaborative autonomous vehicles (CAVs). A graphical representation of the Flourish system is shown in Fig. 25. The objective of applying IoTsecM over the design of the Flourish project is to provide an application architecture where the security mechanisms and controls are depicted and modeled to enable secure, trustworthy, and private technology within the CAVs and the whole infrastructure. Hence, IoTsecM focuses on the security modeling design process for communications and

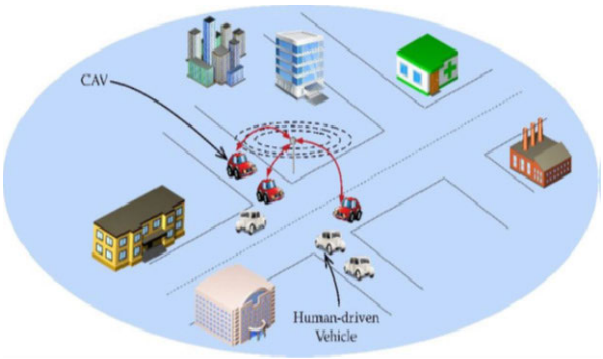


**FIGURE 25.** Graphical representation of the flourish system.

privacy issues. The IoTsecM extensions provide a notation and semantics that model and depict the security concerns in the system architecture model.

There are many threats in the Flourish environment since CAVs will be moving along the city, and then they will be easily reached as well as its communication flow. Therefore, assets may be targeted by numerous cyber-attacks, and probably exposed to many motivated and not motivated attackers.

In general, some of the key security challenges for IoT that also apply to CAVs include: a) naming and identity management, b) interoperability and standardization, c) information privacy, d) objects safety and security, e) data confidentiality and encryption, f) network security and g) spectrum allocation [37].

In the world of CAVs, IoT specifically applies to connect sensors and vehicles to networks [38]–[40]. Hence, data produced by CAVs may pose security challenges for the vehicles and their users. These security concerns may derive from the following sources: physical (e.g., side-channel attacks to crack information), interception (such as man-in-the-middle attacks), abuse (such as unauthorized access to the vehicle), malicious code (generic malicious code affecting the integrated infotainment system), data leaks (e.g., when the vehicle changes owner), among others [37].

In the Flourish project, the main security threats considered are a) loss of control over the system as the result of cyber-attacks; b) damage or loss of technology assets (e.g., loss of data or damage caused by a third party); c) any abuse such as denial of service attack or unauthorized access to systems; d) information leakage or sharing, inadequate design, and planning or lack of adoption of standards; e) failures or malfunctions (e.g., software bugs); and f) information interceptions or network reconnaissance. These security threats are inexorably linked with the IoT, as it is how the components of the Flourish system communicate, exchange data, decide, take actions, and provide services.

In our work, the process followed to perform the threat modeling and security countermeasures analysis and design are based on [28]. However, this process was customized and extended to add countermeasures modeling; this is summarized in the following steps 1) identify the assets; 2) create an IoT system architecture overview; 3) decompose the IoT system; 4) identify threats by constructing an attack tree for each threat; 5) document identified threats; 6) propose countermeasures for each threat; 7) propose a system architecture depicting security countermeasures. A more detailed explanation of this methodology can be found in [21].

The Flourish project involves autonomous vehicles communicating with each other, with human-driven vehicles (HDV) and with roadside units (RSU). This communication is referred to as V2X (vehicle to everything) [40]. The system architecture overview is mainly related to CAVs, which are autonomous vehicles traveling around the city; also, there are people who use the CAVs as a transport medium. The RSU are the communication hubs that are strategically located to communicate CAVs and various processing centers. Therefore, the system architecture overview includes three assets' categories: a) CAV, b) RSU, and c) Processing nodes.

Flourish is a complex ecosystem that consists of many different assets defined for distinct scenarios described by the Flourish team. For the matter of space, those scenarios are not described in this paper; however, the list of identified assets is shown in Table 3. The assets identification allows an understanding of what must be protected. Assets are system components that are of interest to an attacker; therefore, they can be hardware, software, physical entities, or even humans. Assets were obtained by analyzing the scenarios provided by the Flourish team, who described each scenario as general system use cases.

Once the assets are identified, following the threat modeling described earlier, the next step is to create a system architecture overview. The system architecture is depicted in a UML class diagram; it considers the assets identified before and their connections. As shown in Fig. 26, there are three main components within the architecture: CAVs, RSU, and Intelligent Transport Systems (ITS) central station.

The CAVs hold the onboard sensors, the vehicle level AI unit, and the autonomous control; besides, they contain some attributes such as an ID and driven intensions. The operations that the CAVs hold are: feedBBR, receiveInstructions, broad-castDrivenIntensions, broadcastMotionHDV, readManoeuvringActions, provideODinformation, avoidCongestion, receiveRoutingAdvisory and aggregateODInformation. Each of these instructions corresponds to one functionality described in the scenarios, e.g., the broadcastDrivenIntensions operation corresponds to the use case of maneuvering collaboration where CAVs must broadcast their driven intentions to other CAVs for them to react to the new movements correctly and even predict new driven intensions.

The communication channel between CAVs and RSU may use 3G/4G technology and ITS-G5 OBU, which is related to the infrastructure proposed by the Flourish team; these two communication channels allow the CAVs to send and receive data from the RSU.

The RSU operations are described in Table 4. The principal RSU functionality is to receive information from the

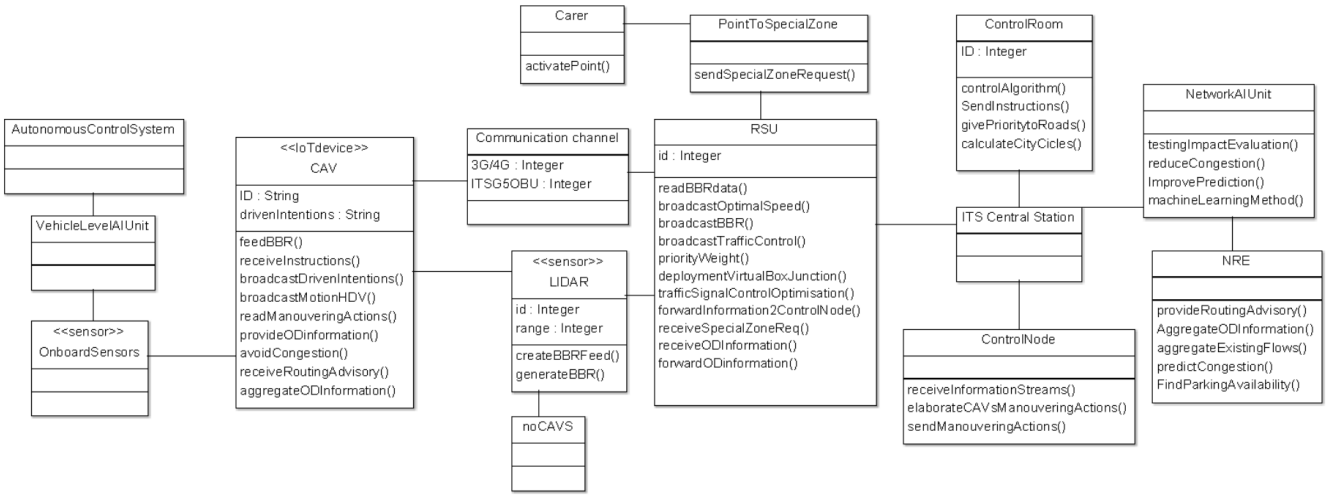


**FIGURE 26.** Flourish architecture overview.

processing nodes and forward data to the CAVs. The operations correspond to the different kinds of data that the RSU must forward.

The ControlNode class models the control node identified as an asset, the operations defined to model the control node behavior are receiveInformationStreams, elaborateCAVsManoeuveringActions, and sendManoeuveringActions. These operations are related to the collaborative maneuvering actions. The actions are calculated in the control node and sent to the RSU for the CAVs to receive them and act accordingly.

The network rules engine asset is modeled with the NRE class, which contains the following operations: provideRoutingAdvisory, aggreagateODinformation, aggregateExistingFlows, predictCongestion, and FindParkingAvailability. The operations defined for the NRE class model define the NRE behavior, such as to find parking availability, predict congestion, etc.

The network AI unit is modeled with the NetworkAIunit class; therefore, its operations correspond to its behavior. The control room is modeled with the ControlRoom and its operations (controlAlgorithm, SendInstructions, givePriorityRoads, and calculateCityCicles) which are focused on prioritizing traffic flow on certain roads and send instructions to the RSU. The special zone point is modeled with the PointToSpecialZone class, which is displayed on the Flourish architecture. The main functionality of this asset is to send the request for a special zone; thus, the sendSpecialZonRequest is proposed to model that behavior. The carer is the person who requests the activation of a special zone through a zone point, this asset is modeled by the Carer class, and it includes one operation activatePoint. LIDAR is the asset that monitors the CAVs and HDV; it obtains and creates data about the vehicle's movements. The operations defined for the LIDAR class are createBBRFeed and generateBBR. NoCAVs is the class that models the HDV vehicles and other moving objects that the LIDAR may observe.

Once the Flourish system architecture is understood, it is time to identify the possible threats. In order to address the threat identification, since the system has not been deployed yet, all the analysis is done based on graphic representations.

Therefore, attack trees diagrams are proposed to address threats identification. Attack trees are an orderly and sequential way of describing the sub-attacks to violate a system; they are useful for conceptualizing and visualizing possible attacks. This analysis identifies the underlying root causes of attacks, allowing the analyst to create attacker profiles, make decisions about the possible mechanisms and security controls needed to protect the system from some attack profiles and thus reducing the attack surface.

Building an attack tree is not an easy job since it must consider the entire attack surface as far as possible. Hence, the use of a tool for this purpose is recommended; in this work SecureItree, built by the Canadian company Amenaza (the Spanish word for threat) [41], was used.

For the case of the Flourish system, the following six possible threats were identified 1) block communication channel from CAVs to RSU, 2) spoofing of BBR data, 3) carer impersonation, 4) jamming of the RSU communication, 5) spoof RSU output data, and 4) flashing control node data. For the matter of space, only the attack trees for the first two threats above are presented, readers interested in the details of the remaining attack trees are referred to [20].

One of the underlying concerns for the Flourish system is the communication flow; hence, the first threat to be modeled is related to the communication between the CAVs and the RSU, as depicted in Fig. 27. The threat identified is named Block communication channel from CAVs to RSU; this would interrupt any communication between those assets, attacking the availability of the system. The sub attacks associated with the success of the main attack are:

- Jamming data from CAVs to RSU: this kind of attack is prevalent to compromise a wireless environment such as the communication between the RSU and CAVs; its goal

**TABLE 3.** Flourish assets.

| Asset | Description |
|---|---|
| LIDAR | It is a sensor located in strategic places and it creates BBR data (data monitored from other cars) |
| CAVs | The collaborative autonomous vehicles |
| RSU | The roadside unit |
| BBR data | Data created by a LIDAR about what it observes |
| BBR+ feed data | The same BBR data plus new observations |
| RSU instructions | Data sent from RSU to CAVs |
| Traffic signal control data | Data sent from the control center to control traffic signals |
| Optimization of traffic signal control data | Optimization of data sent from the control center to control traffic signals |
| Control node | This is the node which determines the CAVs maneuvering actions when information related to HDV intentions and CAVs driven intentions is received |
| CAVs maneuvering actions data | They are the main result of the control node processing unit, they indicate to CAVs the new maneuvering actions |
| Motion description of HDV data | They are the data within the RSU that support the motion description of HDV |
| Carer | It is the person dedicated to activate the point in order to establish a special zone which is a restricted zone for special requirements, such as slow traffic |
| Point to activate especial zone | It is the point dedicated to send the request for a special zone |
| Request for exclusion zone data | It is the request sent, by the carer, through the special point to the RSU to broadcast it to the CAVs |
| Special zone information | This is information about the special zone requested, such as duration, space restrictions, etc. |
| Network Rules Engine (NRE) | It is the processing node dedicated to providing routing advisory to communicate to the Network AI Unit |
| Congestion reduction data | They are data generated from the NRE and related to reduce the congestion on roads |
| Network AI unit | It is the artificial intelligence unit which use machine learning methods to improve the routing advisory for CAVs |
| Machine learning method | It is the machine learning method applied by the Network AI unit |
| Congestion prediction information | It is the result of the machine learning method which predicts the congestion on roads |
| Routing advisory data | They are the data used to advise CAVs with new routes to reduce the traffic |
| Control room | It is another processing node which gives priority to roads |
| Control algorithms | These are the algorithms that are contained within the control room and are used by it |
| Instructions data | They are data delivered by the control room and sent to CAVs through the RSU |
| On board sensors | They are sensors located within the CAVs mainly thought to monitor passengers |
| Vehicle level AI unit | The artificial intelligence unit within the CAVs |
| Autonomous control system | The subsystem which carries out the CAVs control |

**TABLE 4.** RSU operations

| RSU Operations | Description |
|---|---|
| readBBRdata | Read BBR data (data from LIDAR about what it observes) |
| broadcastOptimalSpeed | Communicate optimal speed to CAVs |
| broadcastBBR | Communicate BBR data |
| broadcastTrafficControl | Communicate traffic data from the control center to CAVs |
| priorityWeight | Assigns wights to define the priority of roads |
| deploymentVirtualBoxJunction | Determines a virtual box for a road junction |
| trafficSignalControlOptimisation | Communicate optimized traffic control data |
| forwardInformation2ControlNode | Communicate CAVs maneuvering actions to control nodes |
| receiveSpecialZoneReq | Receive the request for a special zone |
| receiveODinformation | Receive odometry data from CAVs |
| forwardODinformationCAVs | Communicate odometry data to CAVs |

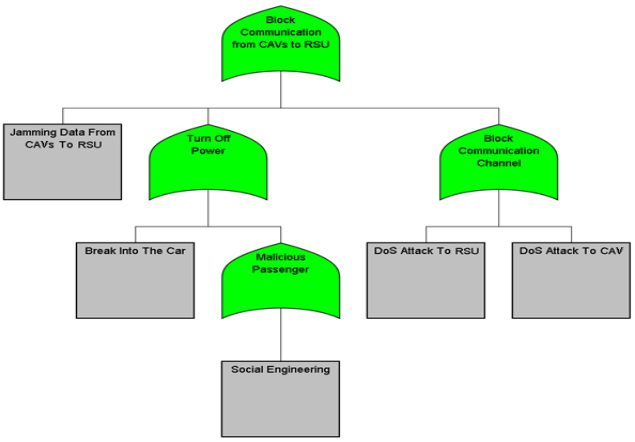


**FIGURE 27.** Block communication from CAVs to RSU attack tree.

is to drop the signal to a level where the communication is interrupted. Typically, old wireless area networks infrastructures are the most vulnerable to this kind of attack since current networks can adapt to unintentional or intentional interference. The countermeasure proposed for this attack is an intrusion prevention system (IPS) since it should detect the presence of any unauthorized client device.

- Turn off power: this attack is related to turn off the power of a CAV, with two ways to achieve it: break into the car, which may be by brute force, in this case, a countermeasure is to enforce the car door locks; the other way is when a malicious passenger gets into the car, for example through

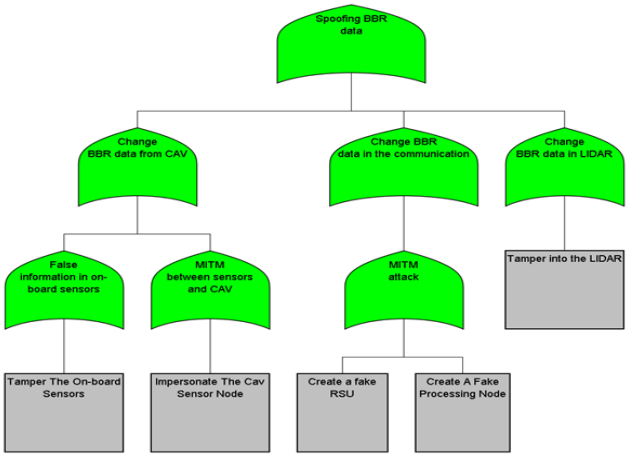


**FIGURE 28.** Spoofing BBR data attack tree.

using social engineering to obtain authorized credentials; in this case, the countermeasure proposed is to enforce the policies to get entry into the CAVs.

- Block communication channel by DoS attack: this attack consists in sending many requests to the CAVs or to the RSU to make them attend just the false and malformed requests, whereas they deny any other request, even the authenticated requests. The countermeasure against this attack is the use of an IDS or an IPS.

The next attack tree presented here considers the spoofing of BBR data, shown in Figure 28. This attack means that an attacker can somehow masquerade himself as another actor to falsify data, in this case, the BBR data generated by the LIDAR sensor. In other words, this threat describes the BBR data falsification. In this case, we are focusing on the leaf attacks since blocking these attacks would thwart the complete vector attack. The attack tree considered for this threat involves the following sub attacks:

- Tamper the onboard sensors: this attack means that the onboard sensors are manipulated by an attacker who can change the data the sensor is reading to create false information and consequently change the BBR data from CAV. The countermeasure for this attack is hardware and software tamper-proof for the on-board sensors; in this way, the attacker who has reached the sensors physically will not be able to perform the tamper attack.
- Impersonate the CAV sensor node: another way to change the BBR data from CAV is through a man in the middle (MITM) attack. If an attacker can impersonate the CAV sensor node, then it can receive and change the onboard sensors data. Therefore, the countermeasure for this attack is the authentication of the onboard sensors, to guarantee that the sensors are whom they claim to be.
- Create a fake RSU: this attack consists in creating a false RSU to perform a MITM attack; in this way, the LIDAR would not be able to identify the false RSU, would trust it and it would share the BBR data. The countermeasure proposed against this attack is the authentication of the RSU. Besides, a trust and reputation scheme would help to mitigate this kind of attack.
- Create a fake processing node: this is quite similar to the previous attack. However, in this case, the MITM attack is deployed between the LIDAR and some of the processing nodes. The countermeasure proposed to mitigate this threat is the authentication of the LIDAR and a trusted processing node.
- Tampering into the LIDAR: the most despicable way to attack the system is to tamper with the LIDAR. This attack means that a well-motivated attacker performs a physical tamper to the LIDAR to change the data about to be sent. In this case, as a countermeasure, tamper protection for the software and hardware is required in the LIDAR.

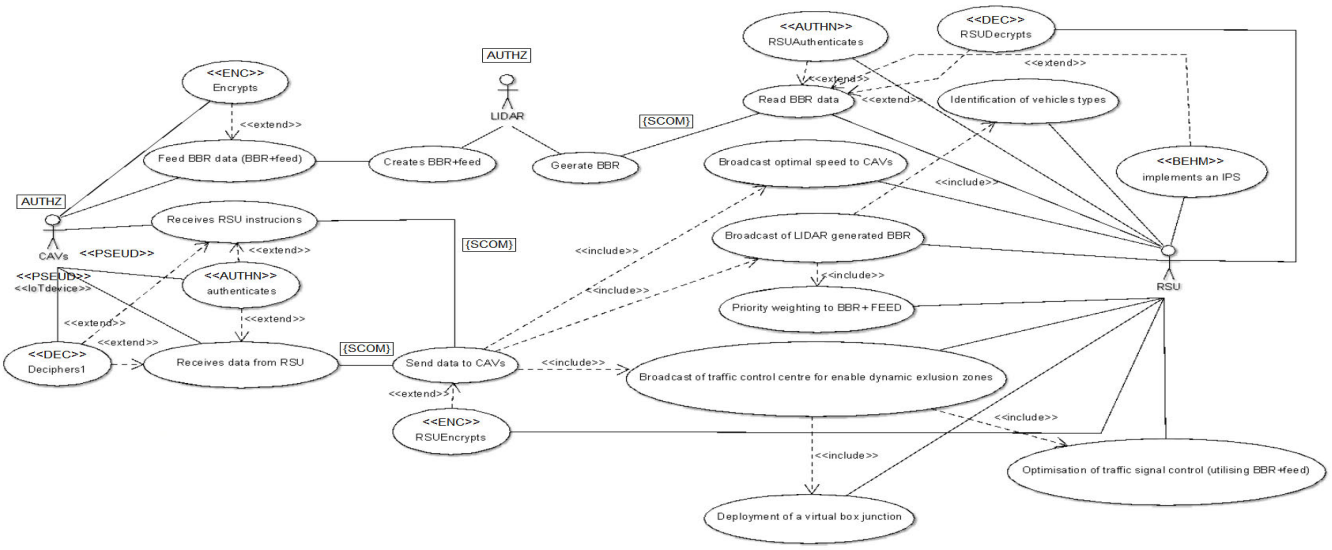


**FIGURE 29.** LIDAR scenario use case diagram.

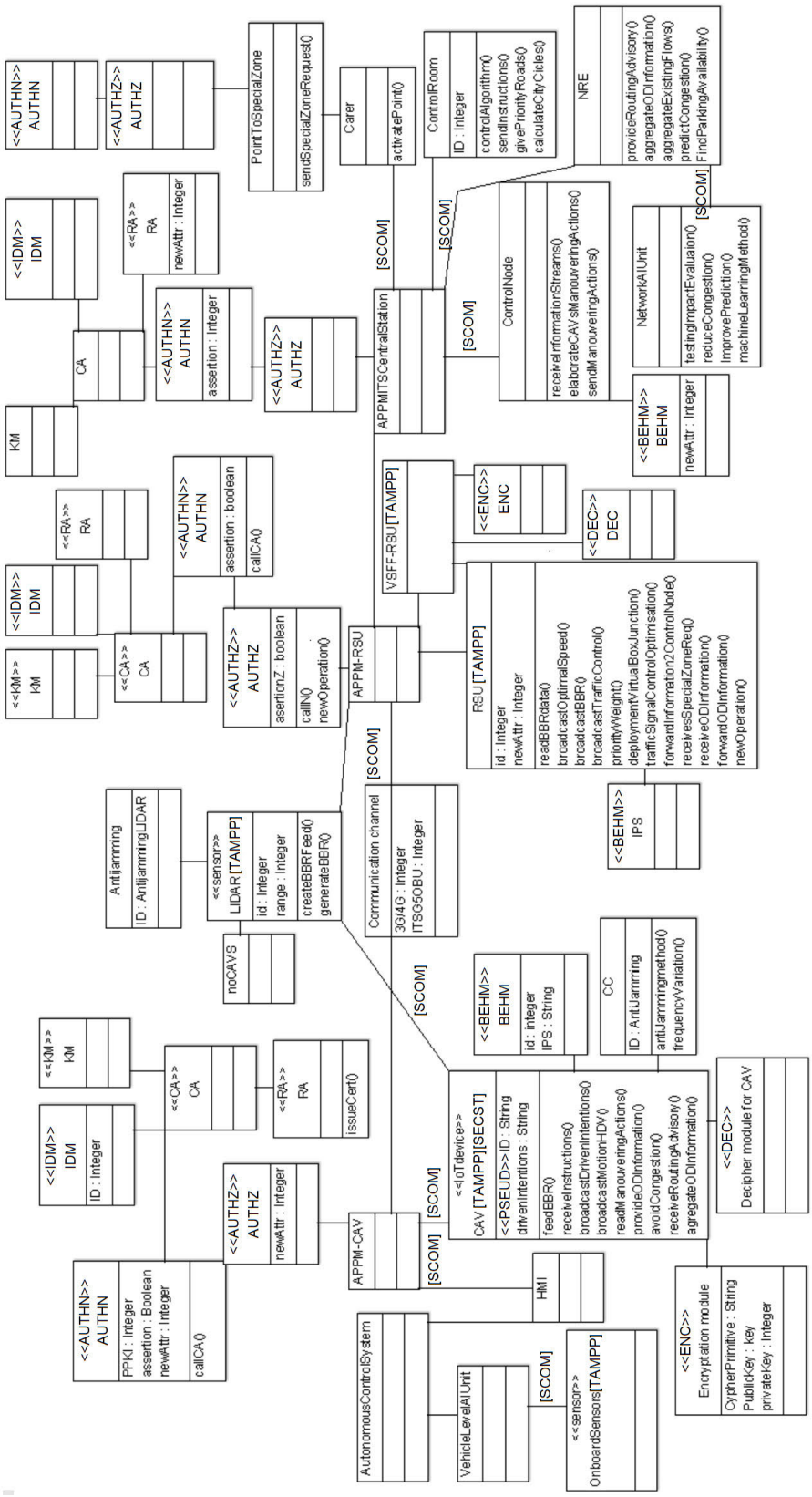


**FIGURE 30.** Flourish architecture applying the IoTsecM profile.

**TABLE 5.** ≪ENC≫ use case for CAV, scenario 1.

| Use case name: <<ENC>> Encrypts |
|---|
| Participating actor: CAVs |
| Entry condition: The CAVs receives data from RSU and authenticate and decrypt them. |
| Events flow: |
| The CAVs actor read the on-board sensors, |
| It obtains the feed BBR data |
| Encrypts them with the RSU public key and send them to the RSU. |
| This use case extends the Feed BBR data (BBR+feed) use case |
| Exit condition: The data package is signed and sent by the CAVs to the RSU |

**TABLE 6.** ≪AUTHN≫ authenticates use case for CAV, scenario 1.

| Use case name: <<AUTHN>> authenticates |
|---|
| Participating actor: CAVs |
| Entry condition: An entry package is sent from RSU |
| Events flow: |
| The package is received. |
| The CAVs actor runs the authentication element. |
| The <<AUTHN>> element obtains the RSU credentials from the package. |
| The <<AUTHN>> stereotype instance creates complementary information from de credentials |
| The <<AUTHN>> stereotype instance runs the authentication function. |
| The <<AUTHN>> creates the assertion {True, False}. |
| This use case extends the Receives RSU instructions use case and Receives data from RSU use case |
| Exit condition: The CAVs authenticate the package received |

**TABLE 7.** ≪DEC≫ deciphers1 use case for CAV, scenario 1.

| Use case name: <<DEC>> Decryps1 |
|---|
| Participating actor: CAVs |
| Entry conditions: |
| True assertion from the <<AUTHN>> stereotype use case (authenticates). |
| The <<DEC>> stereotype instance holds the CAVs private key according to the pseudonym at the moment. |
| Events flow: |
| The <<DEC>> Decrypt1 obtains the private key. |
| The <<DEC>> Decryp1 apply the decryption algorithm. |
| The data from the RSU is in plain text now for the CAVs interpretation. |
| Exit condition: The data decryption was correctly done. |

**TABLE 8.** ≪ENC≫ RSUEncrypts use case for RSU, scenario 1.

| Use case name: <<ENC>> RSUEncrypts |
|---|
| Participating actor: RSU |
| Entry condition: Package ready to send, this means that all the use cases send data to CAVs. |
| Events flow: |
| The RSU package all the data to send to CAVs. |
| The RSU actor use the CAV public key to encrypt the data. |
| The RSU signs the package. |
| The RSU fulfil the package adding its credentials. |
| Exit condition: The encryption process was successfully done. |

Once the attack trees have been analyzed and the countermeasures have been identified, it is time to determine where they must be placed.

The IoTsecM profile includes some extensions to the use cases metaclasses. The first approach to the definition of the system architecture is to identify which system actor carries out the security countermeasures identified before. Therefore, according to the scenarios proposed by the Flourish team, a set of use case diagrams for each scenario adding security countermeasures were developed.

The scenario about the LIDAR and its interactions with the CAV and RSU is depicted in Fig. 29. Due to the matter of space, this is the only use case explained here; for the remaining use cases, the reader is referred to [20].

The use cases concerned with the countermeasures are presented in Tables 5 to 11, providing the following details: use case name, participating actor, entry condition, the flow of events, and exit condition.

Besides the security use cases defined before, other constraints are displayed on the use case diagram to integrate more security concerns within the diagram. The CAVs must be authorized actors; this is depicted with an "AUTHZ" over the actor's head, which indicates an authorization constraint for all the CAVs actor instances. A pseudonym must be assigned to each CAV; this is depicted with the stereotype ≪PSEUD≫ applied to the CAVs actor. This means that the security resolution unit provides pseudonyms certificates. It is common for such certificates to be temporal; hence, the certificates are revoked in a short time to guarantee the privacy of CAVs.

Two links are identified as secure communications constraints: *Receives data from RSU and Sends data to CAVs* where the communication from RSU to the CAVs is established. The other Secure Communication ({SCOM}) constraint appears in the link between the LIDAR and the RSU.

The LIDAR needs to be an authorized actor in order to be able to send data to the RSU. The RSU needs to be authenticated; thus, the "AUTHN" text box is placed over its head.

The analysis of the countermeasures identified allows determining the points where the security mechanisms should be placed. The IoTsecM extensions within the use case diagrams allowed to depict each security countermeasure identified in the attack trees.

The next step is to propose the whole system architecture; here, the functional and non-functional elements are shown

**TABLE 9.** ≪AUTHN≫ use case for CAV, scenario 1.

| Use case name: <<AUTHN>> RSUAuthenticates |
|---|
| Participating actor: RSU |
| Entry condition: Receive data from the LIDAR |
| Events flow: |
| The package is received. |
| The RSU actor runs the authentication element. |
| The <<AUTHN>> element obtains the LIDAR credentials from the package. |
| The <<AUTHN>> stereotype instance create complementary information from de credentials |
| The <<AUTHN>> stereotype instance runs the authentication function. |
| The <<AUTHN>> creates the assertion {True, False}. |
| This use case extends the Read BBR data instructions use case. |
| Exit condition: The RSU authenticates the package received. |

**TABLE 10.** ≪BEHM≫ implements an IPS use case for RSU, scenario 1.

| Use case name: <<BEHM>> implements an IPS |
|---|
| Participating actor: RSU |
| Entry conditions: |
| Receive data from the LIDAR. |
| IPS preconfigured. |
| Events flow: |
| The RSU calls the <<BEHM>> Implements a preconfigured IPS. |
| According to the intrusion detected the IPS reacts. |
| The IPS upgrade the policies in the <<AUTHZ>> element |
| Exit condition: The <<BEHM>> instance is running rightly |

**TABLE 11.** ≪DEC≫ RSUDecrypyts use case for CAV, scenario 1.

| Use case name: <<DEC>> RSUDecrypts |
|---|
| Participating actor: RSU |
| Entry condition: The RSU reads the BBR data. |
| Events flow: |
| The <<DEC>> RSUDecrypts obtains the private key. |
| The <<DEC>> RSUDecrypts applies the decryption algorithm. |
| The BBR data is in plain text now for the CAVs interpretation |
| This use case extends the Read BBR data. |
| Exit condition: The data decryption was correctly done. |

in a class diagram. This helps solve all the issues concerning the interconnections between the assets, the identification of their operations, and the relationships between the security mechanisms.

The IoTsecM profile includes extensions for classes, components, and devices meta-classes, which assist the design of the system architecture. In Fig. 30, the system architecture regarding the security elements is depicted. The objective of the IoTsecM profile is to allow designers to build, model, and depict the security mechanisms together with the functional elements; thus, a complete representation for the system may be conceptualized, and then, the system architecture can be established.

As shown in Fig. 30, the security countermeasures identified before are depicted in the system architecture. The CAV requires tamper protection and secure storage; besides, it requires a pseudonym. As shown in the use case diagrams, the CAV authenticates, authorizes, encrypts, decrypts, and monitors the entry data and network. Hence, the ≪AUTHN≫, ≪ENC≫, ≪DEC≫, ≪BEHM≫ and ≪AUTHZ≫ stereotypes are instantiated.

The addition of other security elements such as CA, RA, IDM, and KM follows the necessity of issuing certificates and the pseudonyms requirements. All these elements conform a PKI, or in the case of the pseudonyms requirement, the pseudonym public key infrastructure (PPKI).

The RSU must contain the security countermeasures. Therefore, the stereotypes instances associated with the RSU are ≪AUTHN≫, ≪ENC≫, ≪DEC≫, ≪BEHM≫ and ≪AUTHZ≫, besides the CAV, the PPKI infrastructure supported by the RSU. Tamper protection is placed as a requirement.

The processing nodes are the Control Node, the NRE, and the Control Room. The security mechanisms for these processing nodes are modeled contained in a central station; the central station contains the stereotype instances ≪AUTHN≫, ≪ENC≫, ≪DEC≫, ≪BEHM≫ and ≪AUTHZ≫.

## V. CONCLUSION

Besides the emergence of new technologies and applications in this area, the continuous evolution of the IoT paradigm has shaped a heterogeneous ecosystem where new technical and operative challenges are present. These challenges are pervasive, touching probably all aspects of systems design, implementation, and operation, including cybersecurity. In this context, experience has demonstrated that considering cybersecurity aspects during the analysis stage in the design process of any system helps to prevent attacks and facilitates changes in the future, resulting in the reduction of expenses and security risks. In this sense, different approaches have aimed to introduce cybersecurity aspects through all phases of system development processes, mainly supported by technologies and tools already in use. UML and SysML have become significant players in modeling but still face limitations to cope with IoT systems' modeling challenges. Therefore, this work discussed an approach referred to as Internet of Things Security Modeling (IoTsecM), which is a UML/SysML extension that applies UML stereotype mechanisms, UML/SysML diagrams and UMLsec stereotypes.

In the CAV system use case presented, IoTsecM allowed us to depict security concerns applying the stereotypes described by the profile. Once the threat analysis was performed, the countermeasures were identified and depicted with the functional requirements in use case diagrams and class diagrams. The UML notation provided a better understanding of

where the security countermeasures need to be placed, which actor is associated with them, and how they are related to other system assets. This architectural view may be extended with behavioral diagrams where the use cases and objects' actions would be depicted to observe their processes, besides their interactions.

Although IoTsecM has been applied to the Flourish project in this work, the profile has also been applied in securing mHealth (mobile health) applications, particularly for the mApp case of study for urgent care management [42].

Hence, IoTsecM represents a useful tool that helps to understand and consider the security of IoT systems during their design stage before they are implemented in physical objects. Furthermore, it is expected that the IoTsecM approach will facilitate the building of security awareness and consideration specifically in IoT ecosystems to address risks to IoT applications in digital sectors beyond smart cities. IoTsecM depicts and models security requirements, and it is a UML/SysML extension; therefore, it is visual and helps in terms of conceptualization and representation of security requirements. Compared with other state-of-the-art approaches, and to the authors' knowledge, there is no other UML/SysML extension covering all these aspects.

## REFERENCES

[1] P. J. Escamilla-Ambrosio, A. Rodríguez-Mota, E. Aguirre-Anaya, R. Acosta-Bermejo, and M. Salinas-Rosales, "Distributing computing in the Internet of Things: Cloud, fog and edge computing overview," in *NEO*, vol. 2016, Y. Maldonado, L. Trujillo, O. Schátze, A. Riccardi, M. Vasile, Eds. Cham, Switzerland: Springer, 2018, pp. 87–115.

[2] S. Agrawal and M. L. Das, "Internet of Things—A paradigm shift of future internet applications," in *Proc. Nirma Univ. Int. Conf. Eng.*, Dec. 2011, pp. 1–7.

[3] R. Kandaswamy and D. Furlonger. (2020). *Blockchain-Based Transformation*. Accessed: Jan. 15, 2020. [Online]. Available: https://www.gartner.com/en/doc/3869696-blockchain-based-transformation-a-gartner-trend-insight-report/

[4] T. A. Ahanger and A. Aljumah, "Internet of Things: A comprehensive study of security issues and defense mechanisms," *IEEE Access*, vol. 7, pp. 11020–11028, 2018, doi: 10.1109/ACCESS.2018.2876939.

[5] D. A. Robles-Ramirez, P. J. Escamilla-Ambrosio, and T. Tryfonas, "IoTsec: UML extension for Internet of Things systems security modelling," in *Proc. Int. Conf. Mechatronics, Electron. Automot. Eng. (ICMEAE)*, Cuernavaca Morelos, Mexico, Nov. 2017, pp. 151–156.

[6] *Flourish*. Accessed: Aug. 5, 2019. [Online]. Available: http://www.flourishmobility.com/

[7] A. Serbanati, A. Salinas-Segura, A. Olivereau, Y. B. Saied, N. Gruschka, D. Gessner, and F. Gomez-Marmol, "Internet of Things architecture IoT," in *Project Deliverable D4.2—Concepts and Solutions for Privacy and Security in the Resolution Infrastructure*, N. Gruschka, D. Gessner, Eds. Luxembourg: Publications Office Eur. Union, 2012.

[8] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed security model and threat taxonomy for the Internet of Things (IoT)," in *Proc. 3rd Int. Conf. Netw. Secur. Appl.* Berlin, Germany: Springer, Jul. 2010, pp. 420–429.

[9] S.-W. Lin, B. Miller, J. Durand, G. Bleakley, A. Chigani, R. Martin, B. Murphy, and M. Crawford, "The industrial Internet of Things volume G1: Reference architecture," Ind. Internet Consortium, Object Manage. Group, Needham, MA, USA, 2019.

[10] S. Schrecker, H. Soroush, J. Molina, J. P. LeBlanc, F. Hirsch, M. Buchheit, A. Ginter, R. Martin, H. Banavara, S. Eswarahally, K. Raman, A. King, Q. Zhang, P. MacKay, and B. Witten, "Industrial Internet of Things, G4: Security framework," Ind. Internet Consortium, Object Manage. Group, Needham, MA, USA, 2016.

[11] T. Eterovic, E. Kaljic, D. Donko, A. Salihbegovic, and S. Ribic, "An Internet of Things visual domain specific modeling language based on UML," in *Proc. Int. Conf. Inf., Commun. Autom. Technol. (ICAT)*, Sarajevo, Bosnia Herzegovina, Oct. 2015, pp. 1–5.

[12] K. Thramboulidis and F. Christoulakis, "UML4IoT—A UML-based approach to exploit IoT in cyber-physical manufacturing systems," *Comput. Ind.*, vol. 82, pp. 259–272, Oct. 2016, doi: 10.1016/j.compind.2016.05.010.

[13] F. Basile, P. Chiacchio, and D. Gerbasio, "On the implementation of industrial automation systems based on PLC," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 990–1003, Oct. 2013.

[14] J. Jârjens, "UMLsec: Extending UML for secure systems development," in *Proc. 5th Int. Conf. Unified Modeling Lang.* Berlin, Germany: Springer, 2002, pp. 412–425.

[15] R. Oates, F. Thom, and G. Herries, "Security-aware, model-based systems engineering with SysML," in *Proc. 1st Int. Symp. ICS SCADA Cyber Secur. Res.*, Leicester, U.K., Sep. 2013, pp. 78–87.

[16] A. Van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proc. 26th Int. Conf. Softw. Eng.*, Edinburgh, U.K., May 2004, pp. 148–157.

[17] A. V. Uzunov, E. B. Fernandez, and K. Falkner, "ASE: A comprehensive pattern-driven security methodology for distributed systems," *Comput. Stand. Interface*, vol. 41, pp. 112–137, Sep. 2015.

[18] L. Apvrille and Y. Roudier, "SysML-Sec: A SysML environment for the design and development of secure embedded systems," in *Proc. Asia–Pacific Council Syst. Eng.*, 2013, pp. 8–11.

[19] L. Apvrille and Y. Roudier, "SysML-Sec attack graphs: Compact representations for complex attacks," in *Proc. Int. Workshop Graph. Models Secur.* Cham, Switzerland: Springer, 2015, pp. 35–49.

[20] D. A. Robles-Ramírez, "IoTsecM: UML/SysML extension for Internet of Things security modeling," M.S. thesis, Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico City, Mexico, Jan. 2018. [Online]. Available: http://www.cic.ipn.mx/~pescamilla/tesis_terminadas/MSc_David_Robles_final.pdf

[21] D. A. Robles-Ramirez, P. J. Escamilla-Ambrosio, R. Acosta-Bermejo, E. Aguirre-Anaya, A. Rodrìguez-Mota and J. J. Reyes-Torres, "Security oriented methodology for designing Internet of Things systems," in *Smart Technology*, F. T. Guerrero, J. Lozoya-Santos, E. G. Mendivil, L. Neira-Tovar, P. R. Flores, J. Martin-Gutierrez, Eds. Cham, Switzerland: Springer, 2018, pp. 96–107.

[22] *Papyrus*. Accessed: Apr. 13, 2018. [Online]. Available: https://www.eclipse.org/papyrus/

[23] M. Bishop, *Computer Security: Art and Science*. Boston, MA, USA: Addison-Wesley, 2003.

[24] B. Menkus, "Understanding the use of passwords," *Comput. Secur.*, vol. 7, no. 2, pp. 132–136, 1988, doi: 10.1016/0167-4048(88)90325-2.

[25] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015, doi: 10.1016/j.comnet.2014.11.008.

[26] P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. De Bruijn, C. De Laat, M. Holdrege, and D. Spence, *AAA Authorization Framework*, document RFC 2904, Internet SoCity, Aug. 2000.

[27] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 2018.

[28] B. Russell and D. van Duren, *Practical Internet of Things Security: Design a Security Framework for an Internet Connected Ecosystem*. Birmingham, U.K.: Packt, 2018.

[29] D. Gambetta, "Can we trust trust?" in *Making Breaking Cooperation Relations, Electronics* (Department of Sociology), D. Gambetta, Ed. Oxford, U.K.: Univ. Oxford, 2013, pp. 213–237. [Online]. Available: http://www.sociology.ox.ac.U.K./papers/gambetta213-237.pdf

[30] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, Maui, HI, USA, Jan. 2000, pp. 1–15.

[31] D. Boneh, D. Lie, P. Lincoln, J. Mitchell, and M. Mitchell, "Hardware support for tamper-resistant and copy-resistant software," Comput. Sci., Stanford University, Tech. Rep. CS-TN-00-97, Aug. 2000.

[32] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas, "AEGIS: Architecture for tamper-evident and tamper-resistant processing," in *Proc. 25th Anniversary Int. Conf. Supercomput. Anniversary*, Munich, Germany, 2014, pp. 357–368.

[33] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, May 2013, doi: 10.1016/j.adhoc.2013.04.014.

[34] B. B. Zarpelão, R. S Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.
[35] A. Shostack, *Threat Modeling: Designing for Security*. Hoboken, NJ, USA: Wiley, 2014.
[36] B. Potter, "Microsoft SDL threat modelling tool," *Netw. Secur.*, vol. 2009, no. 1, pp. 15–18, Jan. 2009.
[37] B. Sheehan, F. Murphy, M. Mullins, and C. Ryan, "Connected and autonomous vehicles: A cyber-risk classification framework," *Transp. Res. A, Policy Pract.*, vol. 124, pp. 523–536, Jun. 2019.
[38] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," in *Proc. Hacking Conf.*, 2016, vol. 24, no. 8, pp. 1–5.
[39] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, "Security requirements for automotive on-board networks," in *Proc. 9th Int. Conf. Intell. Transp. Syst. Telecommun., (ITST)*, Oct. 2009, pp. 641–646.
[40] J. McCarthy, J. Bradburn, D. Williams, R. Piechocki, and K. Hermans *Connected & Autonomous Vehicles: Introducing the Future of Mobility*. Epsom, U.K.: Atkins, 2016.
[41] *Amenaza*. Accessed: Aug. 5, 2019. [Online]. Available: http://www.amenaza.com/
[42] P. J. E. Ambrosio, D. R. Ramírez, S. Alsalamah, T. Tryfonas, S. Orantes Jiménez, A. Rodríguez Mota, S. AlQahtani, T. Nouh, H. Alsalamah, S. Almutawaa, H. Alkabani, M. Alsmari, N. Alashgar, A. Alrajeh, and H. Kurdi, "Securing health applications using loTsecM security modelling: Dentify. Me app case study for urgent care management," *Comput. Sistemas*, vol. 23, no. 4, pp. 1139–1158, Dec. 2019.

**PONCIANO JORGE ESCAMILLA-AMBROSIO** (Senior Member, IEEE) received the B.Sc. degree in mechanical electrical engineering and the M.Sc. degree (Hons.) in electrical engineering from the National Autonomous University of Mexico (UNAM), in 1995 and 2000, respectively, and the Ph.D. degree from The University of Sheffield, U.K., in January 2004. From 2003 to 2010, he was a Research Associate with the University of Bristol, U.K., within the Departments of Aerospace Engineering and Computer Science. From 2010 to 2011, he was a Research Associate with the Department of Electronics, National Institute of Astrophysics Optics and Electronics, Mexico. From 2011 to 2013, he was the General Director of Innovation and Development at the Scientific Division of the Secretariat of the Interior, Mexico. He is currently a Researcher with the Computing Research Centre, National Polytechnic Institute, Mexico. He has more than 80 publications among journals, conference proceedings, and book chapters. His research interests include cybersecurity, security in the IoT and wireless sensor networks, applications of the Internet of Things, wireless sensor networks, and multi-sensor data fusion.

**DAVID ALEJANDRO ROBLES-RAMÍREZ** received the B.Sc. degree in mechatronics engineering and the M.Sc. degree (Hons.) in computer science with cybersecurity specialty from the National Polytechnic Institute (IPN), Mexico, in 2015 and 2018, respectively. He did a research stay at the University of Bristol, U.K., from May 2017 to August 2017, working in the Flourish Project. From 2018 to 2019, he worked as the Internet of Things (IoT) Expert Analyst for AT&T Mexico within the IoT Network Department. In 2019, he worked as the IoT and Data Science Expert with Grupo Salinas, within the Research and Digital Transformation Department. Since 2020, he has been working with Schlumberger, USA, as the IoT and Edge Computing Analyst and Developer. He is currently on a research stay at the Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico.

**THEO TRYFONAS** received the B.Sc. degree in computer science from the University of Crete, Greece, and the M.Sc. degree in information systems and the Ph.D. degree in informatics from The University of Athens.

His Ph.D. was sponsored by Ernst & Young, one of the "Big Four" global assurance and advisory firms. He is currently an Associate Professor of smart cities and urban innovation with a background in systems engineering, cybersecurity, and software development with the University of Bristol. He teaches across undergraduate programs of engineering design and civil engineering and the M.Sc. in computer science. He is a fellow of the Royal Society of Arts and a Chartered Professional Member of the Chartered Institute for IT (MBCS CITP).

**ABRAHAM RODRÍGUEZ-MOTA** received the B.Sc. degree in communications and electronics engineering from the Superior School of Mechanical and Electrical Engineering, Campus Zacatenco Instituto Politécnico Nacional, Mexico, the M.Sc. degree in communications, computer and human centred systems from the University of Birmingham, Birmingham, U.K., and the Ph.D. degree in Computer Science from The University of Sheffield, Sheffield, United Kingdom.

**GINA GALLEGOS-GARCÍA** received the B.Sc. degree in computing engineering, the M.Sc. degree in communications and microelectronics, and the Ph.D. degree in communications and electronics from the National Polytechnic Institute (IPN), Mexico, in 2003, 2005, and 2011, respectively. During the summer of 2011, she performed a postdoctoral research at Yale University, USA. From 2011 to April 2019, she was a Research Associate at the Mechanical and Electrical Engineering School, IPN. She is currently an Associate Researcher with the Computing Research Centre, IPN. She has publications in journals, conference proceedings, and book chapters. Her research interests include formal and practical modern cryptography, post-quantum cryptography in devices of constrained resources, and security in cyberspace and information systems. She is a member of the IACR.

**MOISÉS SALINAS-ROSALES** received the Ph.D. degree in communications and electronics. He is currently a Research Professor with the Computer Research Center, National Polytechnic Institute, Mexico. He worked as an information security specialist in various consulting projects in both public and private organizations. He has also published works in research journals and technical conferences and has served as an advisor to various master's, specialty, and bachelor's thesis. His research interests include information security controls, cryptographic protocols, and secure implementations.