INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

T E S I S

IoTsecM: UML/SysML Extension for Internet of Things Security Modelling

QUE PARA OBTENER EL GRADO DE

Maestría en Ciencias en Ingeniería de Cómputo

P R E S E N T A

Ing. David Alejandro Robles Ramirez

DIRECTOR DE TESIS

Dr. Ponciano Jorge Escamilla Ambrosio

Ciudad de México Diciembre 2017

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 16:00 horas del día 01 del mes de diciembre de 2017 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"IoTsecM: UML/SysML Extension for Internet of Things Security Modelling"

Presentada por el alumno:

Robles	Ramirez	David Alejandro
Apellido paterno	Apellido materno	Nombre(s)

Con registro: B 1 5 1 2 2 4

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director de Tesis

Dr. Ponciano Jorge Escamilla Ambrosio

Dr. Herón Molina Lozano

Dr. Raúl Acosta Bermejo

Dr. Moisés Salinas Rosales

Dr. Abraham Rodríguez Mota

Dra. Sandra Dinora Orantes Jiménez

PRESIDENTE DEL COLEGIO DE PROFESORES

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACION
DIRECCION

Dr. Marco Antonio Ramírez Salinas

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 6 del mes diciembre del año 2017, el (la) que suscribe David Alejandro Robles Ramírez alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Cómputo con número de registro B151224, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Ponciano Jorge Escamilla Ambrosio y cede los derechos del trabajo intitulado IoTsecM: UML/SysML Extension for Internet of Things Security Modelling, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección davrobles28@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

David Alejandro Robles Ramírez

Nombre y firma

RESUMEN

El dominio del Internet de las cosas implica un cambio revolucionario en nuestro entorno, nuevos elementos virtuales se agregan al Internet, los cuales son intitulados las "Cosas". Los objetos físicos son representados como datos digitales a través de las mediciones de los sensores que se envían a una puerta de enlace y viajan por Internet a cualquier otra parte del mundo, donde esos datos se traducen en información y se utilizan para el interés del usuario. Los usuarios no son sólo humanos, hay sistemas donde las "cosas" observan, interactúan y actúan con otras "cosas" sin interacción humana.

A medida que crece la cantidad de entidades conectadas a Internet, la superficie de ataque también crece. Los sistemas del IoT imponen nuevos desafíos para la atención de los requisitos de seguridad, además cada día aparecen más atacantes motivados. A pesar de estos hechos, los requerimientos de seguridad dentro de los sistemas de IoT se toman en cuenta como una idea posterior al diseño del sistema, sin importar que la información manejada por esos sistemas es muy sensible en la mayoría de los casos.

En esta tesis se propone un enfoque denominado IoTsecM. Esta propuesta es una extensión UML / SysML para el modelado de requisitos de seguridad dentro de la etapa de análisis en un ciclo de vida de desarrollo de cascada dentro de un proceso de Ingeniería de Sistemas Basado en Modelos. IoTsecM permite la representación de requisitos de seguridad en dos lenguajes de modelado muy conocidos, UML y SysML. Con la utilización de esta extensión, los desarrolladores de IoT pueden tener en cuenta los requisitos de seguridad desde la etapa de análisis en el proceso de diseño de los sistemas de IoT. Esto significa que el enfoque propuesto permite que los sistemas de IoT se diseñen teniendo en cuenta las posibles amenazas y el análisis de los requisitos de seguridad correspondientes.

La aplicación de IoTsecM en sistemas de la vida real mostró poder representar los requerimientos de seguridad de dichos sistemas. IoTsecM se aplicó en dos casos de estudio, el primero relacionado con vehículos autónomos y el segundo, un sistema en el dominio mHealth, en esos sistemas la aplicación de IoTsecM representó los requisitos de seguridad identificados junto con los elementos de arquitectura del sistema, en ambos casos todas las contramedidas identificadas se representaron con el perfil IoTsecM. Por lo tanto, la aplicación de la propuesta fue verificada ya que se demostró que fue capaz de representar todas las contramedidas a ataques identificadas con el perfil IoTsecM.

La utilización de IoTsecM en sistemas IoT, da como resultado la consideración y representación de requisitos de seguridad en diagramas UML / SysML, por lo que es una extensión UML / SysML que ayuda a los desarrolladores a considerar los requisitos de seguridad desde la etapa de análisis para obtener los mecanismos y controles de seguridad para todo el sistema.

ABSTRACT

The Internet of Things domain implies a revolutionary change in our environment, new virtual elements are aggregated to the Internet, and these are the “Things”. The physical objects are represented as digital data through the sensors measurements which are sent to a gateway and travel along the Internet to any other part of the world, where that data is translated into information and used for the user interest. The users are not only humans, there are systems where the “things” observe, interact, and act on other “things” without human interaction.

As the number of entities connected to the Internet grows, the attack surface grows as well. The IoT systems impose new challenges for security requirements identification and depicting, and more motivated attackers appear every day. Nevertheless these facts, the security requirements within IoT systems are reviewed as an after-thought, even when the information handled by those systems is very sensitive in most cases.

In this thesis work an approach referred to as IoTsecM is proposed. This proposal is an UML/SysML extension for security requirements modelling 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 modelling languages, UML and SysML. With the utilisation of this extension IoT developers are able to take into consideration the security requirements from the analysis stage in the designing process of IoT systems. This means that the proposed approach allows IoT systems to be designed considering possible threats and the corresponding security requirements analysis.

IoTsecM demonstrated to be able to depict the security requirements in IoT real-life systems. IoTsecM was applied in two study cases related to autonomous vehicles and mHealth domains, in those systems IoTsecM was able to represent the security requirements identified within the system architecture elements, in both cases all countermeasures identified were depicted with the IoTsecM profile. Therefore, the proposal was validated since the applicability of IoTsecM was demonstrated.

The utilisation of IoTsecM in IoT systems, helps to enforce the inclusion and representation in the form of UML/SysML diagrams of security requirements, hence, it is UML/SysML extension which helps developers to include security requirements from the analysis stage to the latest system design in order to obtain the security mechanisms and controls for the whole system.

ACKNOWLEDGEMENTS

This thesis is the result of more than two years of work, I would like to thank to Dr. Ponciano Jorge Escamilla Ambrosio for all his time and talks we had along my time doing the masters studies, I would like to thank to Dr. Sandra Dinora Orantes Jiménez for guiding me in my thesis development and for her time and considerations. The Cybersecurity laboratory gave me the knowledge and theoretical bases, I would like to thank each one of the laboratory members. The CIC opened its doors, I would like to thank to the directors and administration people. Finally I would like to thank to CONACyT for sponsor me during my time at CIC.

I would like to say thank you to my parents for showing me my basis and show me the way to follow. I would like to thank to Laura, for her support, orientation and time. To all my family and friends.

DEDICATION

This thesis is not a work done just by me. I dedicate this thesis to my parents, they are the fundamental part of my life, and all my achievements contain their names in them. I would like to dedicate this thesis to Laura my life partner who is always there for me in an unconditional way.

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

LIST OF ACRONYMS

Acronym	Meaning
ABAC	Attribute Based Access Control
ACL	Access Control List
AKE	Authenticated Key Exchange
AP	Access Point
AWS	Amazon Web Services
BDD	Block Definition Diagram
BM	Behaviour Monitor
C	Cipher
CA	Certification Authority
CAV	Collaborative autonomous vehicle
CBC	Cipher Block Chaining
CLI	Command Line Interface
CoAP	Constrained Application Protocol
CPS	Cyber Physical Systems
CSR	Certificate Signing Request
D	Decipher
DoS	Denial of Service
DTLS	Datagram Transport Layer Security
DVI	Disaster Victim Identification
EC	Elliptic Curve
ETSI	European Telecommunications Standards Institue
FMEA	Failure Means and Effect Analysis
FTA	Fault Tree Analysis
HDMI	High Definition Multimedia Interface
HDV	Human Driven Vehicles
IBD	Internal Block Diagram
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFC	Integer Factorisation Cryptography
IIRA	Industrial Internet Reference Architecture
IM	Identity Management
IoT	Internet of things
IoT-A	Internet of Things Architecture
IoTsecM	Internet of Things Security Modelling
IPS	Intrusion Prevention System
IPSO	Internet Protocol for Smart Objects
ITS	Intelligent transport systems
ITU	Internet Engineering Task Force
KM	Key Management
KMS	Key Management Systems

LWM2M	Light Weight Machine to Machine
M2M	Machine-to-Machine
MAC	Message Authentication Code
mApp	Mobile Application
MBSE	Model Based Systems Engineering
MITM	Man In The Middle
MOF	Meta Object Facility
N	Authentication
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
NRE	Network Rules Engine
OCL	Object Constraint Language
OMA	Open Mobile Alliance
OMG	Object Management Group
OO	Object Oriented
OWASP	Open Web Application Security Project
PAP	Policy Administration Point
PDP	Policy Decision Point
PE	Physical Entity
PEP	Policy Enforcement Point
PFS	Perfect Forward Secrecy
PII	Personally Identifiable Information
PKC	Public Key Cryptography
PKI	Public Key Infrastructure
PPKI	Pseudonym Public Key
RA	Registration Authority
RBAC	Role Based Access Control
REST	Representational State Transfer
RFID	Radio-frequency identification
RSA	Rivest Shamir Adelman
RSU	Road Side Unit
SAD	Safety Analysis Diagram
SAML	Security Assertion Markup Language
SC	Secure Communication
SS	Secure Storage
SSL	Secure Socket Layer
SSN	Semantic Sensor Network
T&R	Trust and Reputation
TLS	Transport Layer Security
TP	Tamper Protection
TPM	Tamper Proof Management
UML	Unified Modelling Language
V2X	Vehicles to everything
VE	Virtual Entity
VPN	Virtual Private Network
webApp	Web Application

WSN	Wireless Sensor Network
XSS	Cross Site Scripting
Z	Authorization

1 INTRODUCTION

1.1 BACKGROUND – THE INTERNET OF THINGS

Humanity has experienced many technological changes along the digital and computational age, since the first transistor until the quantum computer (at least in concept) the human evolution has had many changes. Industry, developers and users have used the Internet as a principal medium to establish communication, business relationships and even human relationships. The current Internet is the most powerful World Wide Web which through computers are able to share information in a complex structure, it enables computer communications despite the physical distance between them. People use Internet through computers and mobile devices, most of them work with the same protocols and standards. The Internet itself has many characteristics, constraints and considerations. As in the life, in technology *"the only thing that is constant is the change" Heraclitus*. Therefore, the Internet as it was known has changed, nowadays the Internet has evolved in so many ways, now there are more assets such as sensors, mechatronics systems, cyberphisical systems (CPS), etc. which pass data through the Internet to reach computers, CPS or any other asset. Therefore, the current technologic binding is huge and depicts the concept of Internet of Things (IoT).

1.1.1 IoT definition

Kevin Ashton firstly proposed the concept of IoT in 1999, and he referred the IoT as uniquely identifiable interoperable connected objects with radio-frequency identification (RFID) technology, however, this definition does not cover utterly the IoT spectrum nowadays. Internet of Things (IoT) is a domain that integrates different technological and social fields.

There are many approaches which define the IoT, before delving in a unique definition it would be good to introduce the most relevant ones, these definitions include the most important organisms or authorities in the topic [1]:

- Institute of Electrical and Electronics Engineers (IEEE): A network of items- each embedded with sensors- which are connected to the Internet.
- European Telecommunications Standards Institute (ETSI): Machine-to-Machine (M2M) communications is the communication between two or more entities that do not necessarily need any direct human intervention. M2M services intend to automate decision and communication processes.
- International Telecommunications Union (ITU): Network that is available anywhere, anytime, by anything and anyone.
- Internet Engineering Task Force (IETF): The basic idea is that IoT will connect objects around us (electronic, electrical, and non-electrical) to provide seamless communication and contextual services provided by them. Development of RFID tags, sensors, actuators, mobile phones make it possible to materialize IoT which interact and co-operate each other to make the service better and accessible anytime, from anywhere.
- National Institute of Standards and Technology (NIST): Cyber-physical systems (CPS) – sometimes referred to as the Internet of Things (IoT) – involves connecting smart devices and systems in diverse sectors like transportation, energy, manufacturing and healthcare in fundamentally news ways. Smart Cities/Communities are increasingly adopting CPS/IoT technologies to enhance the efficiency and sustainability of their operation and improve the quality of life.
- OASIS:" System where the Internet is connected to the physical world via ubiquitous sensors".
- W3C: "The Web of Things is essentially about the role of Web technologies to facilitate the development of applications and services for the Internet of Things, i.e., physical objects and their virtual representation. This includes sensors and actuators, as well as physical objects tagged with a bar code or NFC. Some relevant Web technologies include HTTP for accessing RESTful services, and for naming objects as a basis for linked data and rich descriptions, and JavaScript APIs for virtual objects acting as proxies for real-world objects."

Provide a unique definition for IoT is not an easy job, nevertheless, based on the previous definitions it is possible to identify some main points and take them as a benchmark. The IoT is characterised as interworking networks, which incorporate physical objects, through their virtual representation which is normally their physical characteristics obtained and observed by sensors.

These virtual objects (“Things”) state can be modified using actuators. There are many relationships between the IoT things even without human interaction. The IoT uses the current Internet as a communication medium, nonetheless new and lighter protocols have been developed to optimise the IoT devices communications to contra rest their constraints. IoT applications have two main characteristics. The first is the distribution over a large range of processing nodes, and the second is the high heterogeneity of the processing nodes and the protocols they use [2].

There are many application supported by the IoT, they normally involve the next domains: Health-care, Smart Home, Insurance, Automobile/ Transport and Retail [3]. The IoT is not isolated to a particular application sector, on the other hand there is an interconnectivity between the different domains. This affects squarely the IoT systems an application difficulty since now there are many interconnected things even if they belong to different systems or application fields. Therefore, it implies a systems complexity as we have not experienced before, requiring wider communication channels, faster technology, better security mechanisms, etc.

The IoT environment has five major components according with [4]: IoT devices, Coordinator, Sensor Bridge, IoT services, and Controller. This topology helps to see the big picture of the IoT, where new assets are identified such as Sensor Bridge. Therefore, the IoT components are not the same as in the current Internet, the number of assets has grown and, as a consequence, the threats and risks. Hence, there is a wider attack surface to thwart the system functionality.

Although the IoT is certainly the next generation of the Internet, it is still in its infant stages and a lot of technical difficulties associated with its implementation need to be overcome. The success of the IoT relies on the standardisation of security at various levels, which would provide secured interoperability, compatibility, reliability and effectiveness of the operations on a global scale [5].

1.1.2 IoT constraints and challenges

The IoT is a relatively new concept, hence there are still many constraints and challenges to overcome, and as a consequence, there are also many opportunities for developers. The Internet by itself has many constraints and challenges to be overcome, despite of that this section is focused on the IoT-specific challenges, where indirectly the Internet issues might be included, not necessarily in a clumsily way but in a heritage way.

In [6] a challenge compilation was done regarding the IoT systems and the main necessities that they observed, those IoT challenges can be summarised as:

- Availability of internet at everywhere and at no cost: The availability of Internet affects directly to the IoT success, it is seen in [6] that except North America, Europe and Oceania, over 50% of population of the rest of the world still do not have access to the Internet. Hence, the IoT availability is absolutely linked to the Internet availability.
- Security issues: The design and development of the Internet without any security mechanism implementation should be unthinkable since the information within an IoT system is very sensitive. There are many threats around the IoT environment, in particular the challenge is to design proper countermeasures to protect the systems assets and strengthen the current Internet security mechanisms and informatics systems security.
- Low-cost smart sensing system development: Since the IoT will have trillions of sensor nodes around us and in the environment, one of the most important challenges is to develop and fabricate them very cheap.
- Energy: One of the most important challenges is the power supply for sensor nodes. Usually, batteries are used to supply the necessary energy required for sensor signal processing and communication. New environmentally friendly rechargeable batteries need to be developed, due to the high IoT number of devices expected.
- Computational ability: This challenge lies on the previous ones, if the energy and the cost are adequate, then a proper computation power could be achieved, otherwise it could be impossible.
- Scalability: Scalable systems comprise future technologies, protocols, firmware updates, expansion of the system, etc. The scalability is related to other challenges such as availability and energy. For this reason, without a wide communication channel provided by the

infrastructure, the system would be restricted to the number of devices that it can handle. The energy affects thanks to the scalability depends on the device autonomy and the firmware update.

- Power consumption: This challenge is related to the algorithms and communication protocols power consumption, to find the right balance is the real goal. A sensor connected to an IoT device or embedded in the environment, located at a very distant place where human interaction will be hard, need to optimize its algorithms in favour of power consumption.
- Acceptability by the industry, society and government: The IoT systems must be trustworthy, and to guarantee this factor all the challenges referred before must be addressed.

In the middle of the challenge galaxy, researches have been proposing to target each one. This thesis is focused on the security concerns challenge; however it does not live alone, it depends on the battery power, computational power etc. IoT devices have serious constraints in resources and security, so implementing conventional security mechanisms in them is not a simple job, and even in some cases it is unrealizable. A classification of these constraints related to security is introduced below [4]:

Limitations based on hardware.

- Computational and energy constraint
- Memory constraint
- Tamper resistant packaging

Software-based limitations

- Embedded software constraint
- Dynamic Security Patch

Network-Based Limitations

- Mobility

- Scalability
- Multiplicity of devices
- Multiplicity communication medium
- Multi-Protocol Networking
- Dynamic network topology

As it can be seen, there are several challenges that must be addressed in order to provide private companies and governments around the world with a tool to develop more IoT systems. The success of the IoT is entirely dependent on overcoming these challenges. Hence, many researchers have put efforts in order to deal with the IoT challenges.

Before IoT appeared, some industry sectors were not concerned about information systems security or tampering protection, nowadays these industries such as health, vehicles and electro domestics have implemented IoT systems within their devices, hence they are potential targets for cyber-criminals and as a consequence they have to think in security requirements when they develop an IoT system.

1.1.3 Security in the IoT

The IoT is changing many domains such as industries, consumer, commercial technology device owners and infrastructure operators. All these new stakeholders are discovering themselves at the precipice of a security nightmare [7] due to the lack of integrating security requirements in the designing process of IoT systems. Nowadays, IoT is involved in a lot of our everyday devices, but it will present a radical change into the actual Internet when industry and stakeholders put more interest in IoT systems, applications and devices. It is estimated that there are 18 000 millions of devices connected to the Internet [8], number that exceed the people on earth (7 billion people) and it is predicted that in the year 2020 they will be 50 billion of smart devices [8]. This huge number of devices will pose new special constraints and challenges, which must be covered for a functional and secure support of the information, hence the IoT must be able to share and monitor information in real time, complying with strict guidelines on the confidentiality, integrity and availability of user's information. The exigency to comply these security requirements is caused by the essence of the

IoT, which implies the obtaining, handling, distribution and analysis of personal or industrial information without human interaction; in many scenarios, this will represent new vulnerabilities, constraints and threats. IoT has the common Internet security lacks, moreover it also has the sensors, physical and communication security issues. Hence, the implementation of security mechanisms within an IoT system is not an easy job, besides it must be integrated since the analysis and designing stages and not as an after-thought.

Nowadays, IoT systems have many security deficiencies, according to a study done by Hewlett Packard [9] 90% of the IoT systems studied collect at least one piece of personal information through the device, the cloud or its mobile application, so privacy issues need to be considered; 60% of devices have vulnerable interfaces to persistent Cross-site scripting (XSS) and use weak credentials; 80% of applications do not require passwords with sufficient complexity and length; 70% of the devices use unencrypted network service and 70% of the systems analysed allow an attacker to identify valid user accounts. These figures are alarming since the personal information that involves an IoT system exposes not only user information such as passwords, or system credentials, a vulnerable IoT application can directly harm costumers due to the knowledge of routines, health, etc., attackers can retrieve this information and exploit it by some IoT system vulnerability; to provide a context of the risk that could exist if an IoT system is attacked, an attacker might attack an IoT system which controls a bulb at home, so if the attacker gains the right access he would be able to turn on, or turn off that light. On the other hand, considering an IoT system where there is an autonomous vehicle retrieving information from the Internet, if an attacker gains access, then he would be able to control the vehicle and induce the passengers to death, in the worst case. From these examples we can see that violated IoT systems are so perilous since they can turn off a bulb or a human hearth.

The security in IoT applications plays a very important role, it is impossible to propose a unique solution to the security issues within the IoT, nevertheless an analysis of the common faults can lead us to a better understanding of the IoT security requirements. IoT systems, as the informatics systems, must guarantee the 3 security pillars: Confidentiality, Integrity and Availability. However, in the IoT these concepts are wider because there are new actors as sensors and actuators. Therefore, now the IoT stakeholders should think in the system security, human safety, sensor integrity data, actuator authentication, etc. Once the IoT meaning is clear, it is time to think

about the security requirements, and the way to protect the IoT system against the most common attacks.

There are many approaches which consider IoT security since different perspectives, the security goals in the IoT depend on the security requirements of each system, the computational power and the energy supply. Therefore, there are many security requirements within the IoT universe, nonetheless in [10] a table of security requirements is given according to their security infrastructure, the content of the table is shown below:

- System Dependability:
- Service Availability: The user should be able to invoke the service he is authorised to access under all conditions
- Infrastructure Availability: The services provided by the infrastructure should always be available, as their operation is critical to the operation of the IoT. Users should be able to reach the infrastructure. The infrastructure services should be able to operate.
- Infrastructure Integrity: Infrastructure Services should operate properly according to their design.
- Infrastructure Trust: The services provided by the infrastructure Services should be trustworthy.
- Non-repudiation (Service -> User): Services should be accessible to Users that have the right to access them.
- Accountability: Some services could be classified or critical for their provider and could require Users to take responsibility of their action. On the other hand, Users might need providers to take responsibility for the Services they provide, because relying on such Services is critical for them.
- Communication Stack
- Service Layer
- Service Access Control/ Authorisation: Anonymous interactions shall be enabled, as well as group.
- Service Authentication: Users must be able to authenticate Servers.
- Service reputation Metering: As there is a high chance of nodes being compromised due to their physical availability to malicious users, a secondary mechanism for establishing trust is needed.

- Service trust: Users must be able to authenticate.
- Network layer
- Network level Anonymisation: If proper countermeasures are not taken, even Users employing Pseudonyms could be tracked by their network locator.
- Confidentiality: Packet exchange across the IoT is exposed to eavesdropping. In the frame of their proposal (IoT-A) confidentiality is supported through encryption.
- User and Service Privacy
- User privacy protection when using infrastructure: The resolution Component should not be able to derive information about the User or his behaviour from his interaction.
- User privacy protection when using Services: Invoked Service should not be able to derive information about the User or his behaviour from his interaction
- Privacy protection of Service towards User: Users should not be able to derive private information (status, position, ownership, etc.) about the subject providing the service by invoking it.

In this proposal, they split the IoT environment in three categories: System Dependability, Communication stack and User and Service privacy. For each one of these categories they find some requirements related to the Confidentiality, Integrity and Availability, then they propose 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). Nevertheless, the IoT involves more security requirements, for instance in many scenarios a tamper protection is needed or the authorisation not just for the service layer but for the physical layer. In [11] some high level security requirements are given: 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, there are more security considerations for IoT, for instance the different layer requirements, according to a respective IoT architecture.

There are other approaches which have obtained IoT security requirements, some of them depend on the IoT architecture viewing, such as the Industrial Internet Reference Architecture (IIRA) [12]. They have four viewpoints: business viewpoint, usage viewpoint, functional viewpoint and

implementation viewpoint, they define the security according to their viewpoints. For the business viewpoint, it is focused on the return on investment for security, in the context of other considerations such as performance or consumer satisfaction. For the usage viewpoint, they propose: security monitoring, security auditing and security policy management and cryptographic. For the functional viewpoint, it includes common security functions: security audit, identify verification, cryptographic support, data protection and privacy, authentication and identity management and physical protection. In the document the authors mentioned that the previous common security functions contribute to various system security capabilities: secured booting, enhanced trust, enhanced privacy, early attack detection, secure management and automatic threat. For the implementation viewpoint, they outline common security issues: end-to-end security (protected device-to-device communications, confidentiality and privacy of data collected, remote security management and monitoring). In order to achieve those security issues, the authors propose the security by design: securing legacy systems (security gateways), security for architectural patterns (in the three-tier architecture the end-points, information exchange, management and control and data distribution and storage), end-point security (embed security capabilities and policy enforcement directly) and information exchange security (authenticity, confidentiality, integrity and non-repudiation of the communication).

IIRA addresses security requirements specifically for: endpoint security, communication security, management and monitoring security and data distribution security. They describe the next security issues to address in end-points:

- Secure boot attestation: Start from a known secure state, following only a prescribed boot sequence of steps, with no modification of intended execution function.
    - Remote attestation to the integrity of the boot sequence
    - Policy to describe how to proceed when deviation from expected boot sequence is detected.
- Separation of security agent
    - Process-based security agent: The security agent resides in a process.
    - Container-based security agent: Secured container within the end-point. Hardware and software-enforced boundaries.
    - Virtualisation-based security agent: Hypervisors in virtualised environment (firewall, on-demand Virtual Private Network (VPN) connections, mutual

authentication, communication authorisation, data attestation, Intrusion Detection system (IDS), Intrusion Prevention System (IPS), etc.).

  - Gateway-based security agent: When security cannot be added to an end-point an extern security Gateway is added.

- Endpoint identity: Endpoints and other controllable assets must have a unique identity, so they can be managed and tracked via the secure agent. Ideally this identity is hardware-embedded so that it cannot be altered.
- Endpoint attack response: It should defend itself, report the attack and reconfigure itself to thwart the attack based on policy.
- Remote policy management: Defines the configuration of the security controls and functions as a form of a security policy for each endpoint.
- Logging and event monitoring: The security agent monitors and records events as they occur at the endpoint including events pertinent to security violation, user login/logout, data access, configuration update, application execution and communication.
- Application whitelisting: Only known and authorised application code (whitelist). The security management point should update the whitelist.
- Network whitelisting: Only a defined set of source/destination, port and protocol tuples is allowed to communicate to/from the endpoint. The security management system may update the network whitelist.
- Endpoint and configuration control to prevent unauthorised change to the endpoints.
- Dynamically deployed countermeasures: Deploy trusted new countermeasures and other mitigating controls as part of the endpoint security policy.
- Remote and automated endpoint update: The security management system must be able to remotely update the endpoint with trusted software updates via the secure agent through an automated and secure process.
- Policy orchestration across multiple endpoints: Coordination of security policy across multiple endpoints to enable secure, trusted operation workflow across the endpoints
- Peripheral devices management: Security policy concerning whether to allow a peripheral to be connected to or disconnected from the endpoint.
- Endpoint storage management: Data storage and file system at an endpoint must be managed based on security policy. The security management function includes file integrity monitoring, file reputation tracking, data, file, file system or device-level encryption, file and

data access right management, remote access to file system, data loss prevention, and alerting policy violations reporting.

- Access control: network access to endpoints must be controlled based on security policy that allows connections required by the operations and deny all other connections.
- For the communication security, they regard the next points:
- Architectural considerations for information exchange security: When designing security solutions, consideration must be given to requirements in confidentiality, integrity, availability, scalability, resilience, interoperability and performance for both transport layers.
- Security in request-response and publish-subscribe communications: As the protocols of this pattern vary in degrees of support for security, they should be independently and carefully evaluated with regard to confidentiality, integrity and availability requirements.
- Mutual authentication between endpoints: The security policy may specify the acceptable authentication protocols and credentials to be used for authentication. Light-weight protocols may be implemented in resource-constraint devices.
- Communication authorisation: Before granting access of any resource, authorisation must be performed at the endpoint according to the security policy.
- Identity proxy/consolidation point: In the case when the endpoint and their authentications cannot be achieved, these components may be proxied by an endpoint with capability that meets the higher standard and capable of performing the proxy functions (security gateway).
- User authentication and authorisation: User credentials can be used to identify the user of the endpoint uniquely. The combination of endpoint and user access control can be used to present a unique access profile to request access to resources at an endpoint.
- Encryption communication: Data exchange between endpoints over communication channels must be encrypted with cryptographic keys of security strength and cipher suite meeting the security policy requirements.

Management and Monitoring security involves these areas:

- Identity management: Hardware-backed identity is required to determine the identity of the endpoint authoritatively. Keys and certificates should be stored in a hardware-secured container. These hardware-secured containers generate asymmetric key pairs on chip and never

expose private keys outside of the container. They perform crypto-operations on-chip with the private-keys. They export public keys to be distributed to other endpoint or likely signed into PKI certificates by a certificate authority. These containers also perform other crypto-based security operations such as attestation, signing and sealing on-chip.

- Provisioning and commissioning: An endpoint must be provisioned and commissioned securely before it is allowed to participate in the system. This requires identity and credentials to be generated, distributed and installed in the endpoints and registered with the security management system.
- Security policy management: Policy creation, assignment and distribution. Policies are defined on a security management system and communicated to endpoints via the secure agent.
- Endpoint activation management: It is the event where a new endpoint is recognised, authenticated and then permitted to exchange data with other endpoints in the system.
- Credential management: Its lifecycle consists of credential provisioning/enrolment/recognition, additional credential generation (particularly for temporary credentials), credential update and credential revocation/de-recognition.
- Management console: It allows a human user interaction with the security management system.
- Situational awareness: Maintain awareness of situations in the network of endpoints in addition patterns emerging a sequence of events can contribute to an understanding of the current environment of the system.
- Remote update: Mechanism must be in place to automatically, securely and remotely update software/firmware.
- Management and monitoring resiliency: Security management needs to manage and monitor the system network especially when facing non-optimal network conditions such as when it is under attack, degraded or damaged.

For data distribution and secure storage, they consider:

- Data security: In the case of data storage, sensitive data can be protected by employing data encryption at the field, record, file, directory, file system or storage device level.
- Data Centric Policies: They include data security, privacy, integrity and ownership.
- Data Analysis and privacy: Data access policy may be provided to enforce fine-grained data access rules.

- IT systems and the cloud: To protect the data, provenance information and privacy requirements should be attached to it so that ownership and the custody chain for data records can be maintained.

The work done in IIRA describes many security requirements in IoT systems and some proposed architecture elements to target them. This work is very helpful because their perception of the IoT is wider than other approaches, nevertheless some security requirements identified are similar between them or with other approaches. Therefore, an analysis of each one of the requirements offered by these approaches was done, in order to obtain general and encapsulated security issues. Now, other approaches will be introduced to continue with the IoT security review.

Secure design requires establishing the relevant security concerns for endpoints, the communication between them, the management of both the endpoints and the communication mechanisms, and for processing and storing data.

If a system has flaws from its design, they can be exploited by attackers who will be able to penetrate the system just applying common attacks using common tools. Therefore, the idea of reducing the attack surface by design make sense since it will prevent attackers to easily infringe the IoT system.

There are many domains in the IoT ecosystem, it would be impossible to propose a unique IoT security model to cover all the IoT security requirements, for each one of these domains, however a meta-model for IoT systems security would be able to help the IoT stakeholders to model their particular IoT system with the appropriate IoT security requirements abstraction, it should be time invariant and technology-independent.

Security in an IoT system should not be considered as an afterthought, it must be reviewed from the planning and modelling stages of the system, that is, it must be an essential requirement in any IoT system. The common aspects of security in IoT systems must be considered, for instance, integrity of data and trust in services offering the data is crucial; furthermore, confidentiality of data and privacy of users must be ensured. Another vital functionality of IoT is availability [13], this security requirement is related to cybersecurity, but in IoT systems there are more assets. Therefore, is more difficult to provide solutions to the availability requirement. In IoT there are other security issues to consider, besides of the Internet security concerns, such as the physical security of the hardware devices, sensors, actuators, etc. The sensors are in contact with the physical entity.

Therefore, in many scenarios the system interacts with them in a remote way, meaning that the sensors are completely vulnerable, if there is not a countermeasure applied.

Hence, in IoT trusted devices, systems, infrastructure, users and applications are needed [14]. There are many proposals for IoT security, including frameworks such as the one described in [15] where also an architecture is proposed describing three main agents: IoT Devices, IoT Broker and IoT Certificate authority.

Since security is a fundamental fact in IoT systems, it has to be considered from the design and modelling stages, for which there is no protocol, standard or tool available. Hence, the idea of having security requirements considered within the system before implementing it is fundamental to build a system without vulnerabilities and reliable. This model will comply with basic security aspects such as availability, confidentiality and integrity, which will help to implement controls and specific security mechanisms within each of the modules proposed in the abstract system.

This work focuses in the security conceptualisation within the IoT security design process, specifically in the analysis stage, since the security concerns within an IoT system should not be regarded as a final stage or when the system is about to be deployed as it is mentioned in [16]. Usually the idea of a model-based development is to build a model of a system with the aim to be as close to human intuition as possible, this is the case for almost every time that an abstract representation of the system functionality is obtained [17]. This system abstraction is generally developed applying modelling languages instead of a natural language description, modelling languages pursue a better understanding of the system. Therefore, those languages should provide the security analysts and developers with a well-formed language to depict the system security concerns in order to be considered from the IoT system conception.

Until now, we know the IoT environment, the security concerns within it and the necessity of a model-driven approach which regards those security issues in to a system model. As it was mentioned before the modelling languages are a very useful tool to abstract the system characteristics along the analysis, design, and implementation stages of a lifecycle. Therefore, there is need for a modelling language, modelling language extension or an abstraction mechanism to abstract the security requirements within a human intuitive representation together with the system model.

1.1.4 Model Based Systems Engineering and Lifecycle Development models

Model Based systems engineering (MBSE) refers to the practise of simplifying concepts and relationships within graphical or mathematical models, it is a system abstraction which removes unnecessary components in order to facilitate the understanding, to aid decision making, to examine “what if” questions and control and predict events. MBSE replaces the document centric approach which in several cases is still used in the industry. MBSE makes models instead of documents, in that way the development teams can communicate and asses the impacts of changes to the system.

There are several methodologies for the MBSE approach. In [18] a survey of such methodologies can be found. The authors from [18] define methodology as a “collection of related processes, methods and tool. A methodology is essentially a recipe and can be thought of as the application of related processes, methods, and tools to a class of problems that all have something in common”.

There are three main lifecycles development models, these are:

- Royce’s waterfall model,
- Boehm’s spiral model
- Forsberg and Mog’s “Vee” model

Those three models are considered as the seminal model, on them several other lifecycle development models are based. In Fig. 1 the stages of a waterfall model is depicted, these stages are: assumptions, requirements, analysis, design and prototyping. The IoTsecM approach is designed to contribute in the requirements and analysis stages, where the system conceptualisation is achieved, and then the first scenarios and use cases are identified. Therefore, the IoTsecM proposal helps for the security requirements identification and depicting within the analysis stage, since the system design can be modelled.

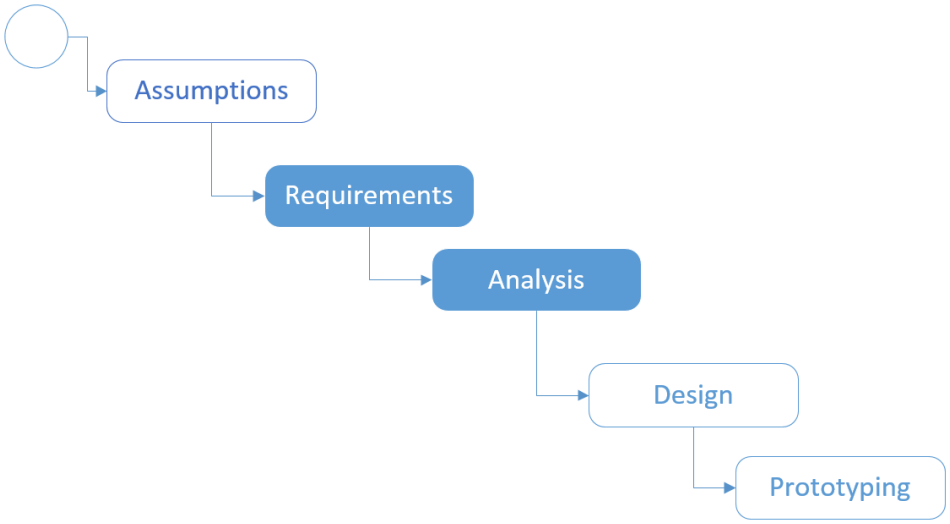

Fig. 1 General Waterfall Lifecycle Development Model

1.2 PROBLEM STATEMENT

Security in IoT systems involves not only stolen information or encryption issues, there are many new assets involved or even a blend of many domains assets of different industries that did not care about information security before; for example, electro domestic industry, where they produced new connected devices, but they did not consider information security in many cases. Therefore, IoT has new challenges never seen before such as new attack surfaces, new scenarios and new attackers that will exploit any vulnerability they find.

With the increasing number of stakeholders involved in IoT systems the attack surface is growing up very fast, new motivated attackers will appear or are waiting the right moment to thwart the different applications and systems in IoT. In IoT systems there are not just involved virtual information or money, there are many new exploitable targets, such as health, houses or any “thing” connected to the Internet. It implies new risks such as lights turned on without authorisation, open doors, turn off cars, set on car false calls, calling to medical services, etc. The universe of possible attacks is wider every day thus it is need new techniques in order to include security in the IoT systems and applications. Although these techniques have to consider security from the analysis stage, they have to adopt systems already built and be able to establish security according to security requirements and threat analysis.

IoT involves experts in many areas from microcontrollers programmers to big data analysts, so the designing process is not an easy job. In addition, there is not a security standard for IoT systems such as ISO 27000 in informatics systems. There is a lack of standard protocols, methodologies and models that supports all features that IoT systems require. One reason to explain it, is that the IoT is a relatively new concept. Therefore, the efforts at this moment have been focused on the new technology to realize the IoT but not much efforts have been directed towards developing security tools to ensure the IoT environment.

It is known that security has always been indispensable in information systems, so it has been covered from many angles, nevertheless the IoT involves new challenges in security requirements, new threats, new risks and vulnerabilities that are always emerging. IoT developers normally are focused on the system functionality without considering security aspects in any process of the methodology (if it is the case) they are following. This can be for many reasons, for example IoT developers usually are not experts in security, there is not a methodology totally oriented to establish security in IoT systems and there is not modelling languages dedicated to security in IoT systems. IoT has new actors involved as: sensors, actuators, devices, networks of devices, IoT brokers, networks of sensors, etc.; hence, new processes for modelling them are essentials for building new IoT systems.

The modelling process would be unthinkable without a modelling language. As it is known the modelling language by default is the Unified Modelling Language (UML) which is very intuitive for the human perception. Nowadays, there are some UML extensions which let designers cover security requirements for information systems or even mechatronic systems but there is not any UML extension for security in IoT systems or applications which can help designers and developers to model the IoT system and the security controls and mechanisms within the architecture. So the lack of IoT modelling with security is a problem because it propitiates security deficiencies from a designing stage and even when the systems is already built.

Therefore, the problem identified is that the IoT systems need a new artifact to allow an abstract representation of the security requirements since the analysis stage in a waterfall development life cycle. UML and SysML are general modelling languages, however, currently they are not able to depict the security domain-specific features, and therefore the proposed new artefact could be a new UML/SysML extension to depict and model IoT security concerns within the analysis stage.

Nowadays there are some UML/SysML extensions which let developers cover security requirements for information systems, in particular in a design and deployment stages, nevertheless there is not any UML/SysML extension to analyse IoT systems since an analysis stage within a Model Based System Engineering approach (MBSE) based on a waterfall model development life cycle. Therefore, the developers cannot model or depict the security mechanisms in the analysis stage of a MBSE approach for each one of the architecture layers in the IoT system.

1.1. Justification

The emergence of a common security reference model for IoT systems can lead to a faster, more focused development and to a better human perception of the trustworthiness of these systems. These solutions can provide many advantages such as economics, new business models which can leverage those technological solutions providing room for economic development.

This security analysis profile will permit the analysis in advance of necessities, vulnerabilities, attacks and countermeasures, including the security relevant requirements tracking, architectures and patterns. All the IoT security features included in a UML/SysML profile will help to mitigate the security design vulnerabilities and also to regard the security requirements within the very well-known UML/SysML diagrams.

Hence, the proposal of this work is a UML/SysML profile that can be applied in any UML/SysML tool. This profile is not just for the security concerns analysis, but it supplies traceable links to the requirements, architectures, designs, models and code.

As it was mentioned before security is essential in IoT systems and new modelling strategies have to be developed in order to cover all IoT security requirements. Although there are many new approaches to model IoT and also new frameworks for security in IoT, there is not a syntax that helps to aggregate security aspects to IoT modelling. This lack of security modelling tools for IoT systems is the main concern of this thesis work. As a result of the research carried out, an UML/SysML extension has been proposed that support IoT designers and developers during security requirements consideration in their systems even if they are not experts in security topics.

Modelling languages have been used by methodologies, processes, methods, etc., to design and enhance information systems in a systematic and ordered way. Among the different modelling languages available, UML is the most used one, it is very intuitive, and it has a simple syntax which makes it the most used modelling language. According to [19], models have views and diagrams to show subsets of engineering data; good diagrams have singular purposes and answer questions, models have scope, purpose accuracy and fidelity, besides models are verifiable and they are engineering data. Hence, an UML extension must comply with that set of features, so it will support good modelling, this is the development of a semantically correct set of engineering data of relevant systems and their properties.

An UML extension for model security requirements in an IoT modelling process will permit designers develop a security layer in their systems and consider security requirements that actual security extensions such as [20] do not cover, for instance sensor authentication, or actuator authorisation, IoT device tampering protection, etc. This security issues are very specific to IoT systems and applications. Therefore, an UML extension is designed to guarantee the inclusion of security requirements along different IoT tiers according to different IoT architectures, besides it also intends to model aspects related to the Internet security.

Within the specification of UML 2 [21] there are some reasons why UML can be extended:

- To give a terminology that is adapted to a particular platform or domain (for example specific terminology like Home interfaces, Enterprise Java Beans, and Archives).
- To give a syntax for construct concepts that do not have a notation (such as in the case of Actions).
- To give a different notation for already existing symbols (such as being able to use a picture of a computer instead of the ordinary Node symbol to represent a computer in a network).
- To add additional semantics to UML or specific meta-classes.
- To add stereotypes that do not exist in UML (such as defining a timer, clock, or continuous time).
- To add constraints that restrict the way UML's constructs are used (such as disallowing multiple inheritance).

1.3 HYPOTHESIS

A UML/SysML extension for IoT security requirements analysis and modelling, will allow the analysts to depict the security concerns from the start of the Model Based System Engineering process based on the waterfall development life cycle. Thus, the countermeasures to protect the IoT system assets will be graphically represented and modelled. This UML/SysML extension will allow to the already built systems to be modelled adding the security mechanisms. This extension will cover the gap between the human conceptualisation and automation models.

1.4 OBJECTIVES

1.4.1 General objective

To design an UML/SysML extension for the Internet of Things systems security requirements depicting, within the analysis stage in a Model Based System Engineering (MBSE) methodology, in particular on such stages in the waterfall development life cycle. This will be achieved applying the existing UML extension mechanisms such as stereotypes, constraints and tags, besides of proposing a nomenclature which aims at fulfilling the IoT security requirements. The UML/SysML extension will allow the analysts to include and represent security countermeasures against possible threats since the designing of any IoT system.

1.4.2 Particular objectives

- To review the IoT UM/SysML extensions.
- To analyse the general IoT security requirements.
- To review the UML/SysML extensions state of the art for IoT security.
- To propose a suitable nomenclature to depict in an abstract way the IoT security requirements.
- To design an UML/SysML extension according to the nomenclature and the existing diagrams as well as the UML extension mechanisms.

- To apply the defined UML/SysML extension mechanisms to extend the UML/SysML semantics to represent the IoT security requirements.
- To propose a UML/SysML extension for IoT security requirements depicting.
- To verify the applicability of the UML/SysML extension.

1.5 SCOPE OF WORK

This research work addresses the design of an UML/SysML extension, which covers the IoT security requirements. The proposal described along this thesis aims at depicting the security requirements within the analysis stage in a waterfall development life cycle of a MBSE approach. This means that the limit for this research work are the security requirements for IoT systems and UML/SysML notation. The UML/SysML extension mechanisms were applied in order to design a profile to depict the security requirements within the analysis stage. It does note, directly, overcome the security issues for other domains or technologies, nevertheless it may be applied in other domains if the developers find it functional. Although this proposal is not, necessarily, applicable for other stages out of the analysis stage in the waterfall development life cycle, or any other development life cycle, it can be applied in other models, methods or life cycles. The IoTsecM proposal was applied in two study cases, hence the applicability of IoTsecM depends on the requirements of future developers and their modelling and analysis skills.

1.6 CONTRIBUTIONS

The following list summarizes the main contributions of this work:

- IoT security requirements compilation and analysis.
- A nomenclature which encapsulates and abstracts the IoT security requirements.
- An UML/SysML extension for security modelling in IoT systems.
- Application of the proposed UML/SysML extension within two real-life projects.

1.7 RESEARCH AND DEVELOPMENT METHOD USED

In this research, in order to obtain the UML/SysML extension a documental research method was followed, since a suitable compilation of the security requirements within the IoT was realised, then that information was analysed and synthetized, it is evident that a deduction process preceded it. Then an experimental method was applied, once the UML/SysML extension was already finalised, it was exposed to two main experiments (real-life experiments) where the extension functionality and applicability were verified, this means that the hypothesis proposed at the beginning was corroborated.

1.8 ORGANISATION OF THE THESIS

In chapter 1 a brief background is introduced which includes IoT definition, IoT constraints and challenges and security in IoT. The problem statement, hypothesis, objectives, scope of the work, contributions and organisation of the thesis sections are included in the chapter 1 as well. In chapter 2 a revision of the state of the art is introduced, in particular a review of the modelling languages specifically designed for IoT systems and UML extension for security concerns modelling. The theoretical framework is described in chapter 3, where the UML is generally described as well as the UML extensions. In chapter 4 the methodology followed, and the research carried out in order to develop an UML/SysML extension for security requirements modelling is presented. This includes a description of the actors, the nomenclature proposed and the extended diagrams. Once the extension has been explained, in the chapter 5 two study cases applying the developed UML/SysML extension are presented. In chapter 6 the conclusions and future work and research outputs are described.

2 STATE OF THE ART

In this section the state of the art is introduced, meaning that the last proposed works related to the thesis topic are addressed and explained. Hence the intention of this chapter is to provide a solid base for the research development, also as to introduce the different perspectives in which IoT systems and their security requirements have been modeled. Once the main approaches have been reviewed, a new way to model the security concerns within IoT systems can be proposed.

The structure of this chapter is split in two main subchapters; the first one is entitled “IoT modelling”, corresponding to section 2.1, which discloses the current UML extension to model IoT systems; the second part, section 2.2, introduces security related UML extensions.

2.1 IoT MODELLING

2.1.1 UML4IoT

The approach named UML4IoT [22] integrates trends in Cyber Physical Systems (CPS) and IoT. They adopt the Open Mobile Alliance (OMA) LWM2M (Light Weight Machine to Machine) application protocol [23], and the smart objects defined by the Internet Protocol for Smart Objects (IPSO) Alliance [24].

The motivation of their proposal is that the automation engineers are not necessarily involved with the Representational State Transfer (REST) architectural paradigm and the LWM2M protocol which the authors claim are developed for IoT systems. Their approach automates the generation process of the *IoTwrapper* which is a transformation of the cyber-physical component to an IoT-compliant component, i.e., to an Industrial Automation Thing. The IoT wrapper is a layer that converts the Object Oriented (OO) API to a RESTful one. Therefore, they propose the automation of that layer. In order to attain that goal they followed two methods, the first one is based on the UML design specification of the cyber-physical component and the second one is based on the source code; we will focus on the first one since it is more related to this work.

The UML4IoT approach has three main contributions:

- The definition of a UML profile for the exploitation of IoT.
- The automation of the generation process of the cyber-physical system IoTwrapper.
- A lightweight prototype implementation of the OMA LWM2M protocol based on metaprogramming.

The whole LWM2M can be found in [23], nevertheless some general concepts need to be introduced. The LWM2M is an application layer communication protocol which decouple system components adopting a plug and play approach. LWM2M protocol defines the LWM2M server and LWM2M client which is located in a LWM2M device. It is defined on top of the Constrained Application Protocol (CoAP) [25] and the datagram transport layer security (DTLS) may be used when it is required. The OMA LWM2M defines the protocol between the LWM2M client and the LWM2M server, it defines five interfaces between them:

- Bootstrap
- Client registration
- Device management
- Service enablement
- Information reporting

Once these concepts were introduced, we can delve more into the UML4IoT approach. The authors aim to embed the LWM2M client in the cyber-physical component, this in order to support general functionalities such as discovery and registration. In their work the authors indicated that an LWM2M object model is not appropriate to define the cyber-physical component structure, thus it is normally modelled applying UML. As it was mentioned earlier, the authors proposed a UML profile to map a UML representation to a LWM2M compliant REST interface allowing the task automation, allowing the developers to design the cyber-physical component using UML. In Fig. 2 the UML4IoT profile core part is shown. The fields and operations of the cyber-physical component are handled as resources which are defined as stereotypes (the meaning of the extension mechanism named stereotype is explained in section 3.2.1); the resource specialisations such as *OperationResource*, *Instance Resource* and *ObservableResource* are used to mark information of the cyber-physical component. They extended meta-classes, such as *Class*, *Property* and *Operation*.

The resources on the OMA LWM2M are organised into objects, using the object type to define the logical organisation of resources, in order to depict this concept they use the *ObcjectType* and *Resource* stereotypes as well as their associations, as it is captured in Fig. 2. The *ObjectType* is defined as a composition of LWM2M resources and it is thought as the cyber-physical component cyber part. The resources are modelled by the *Resource* stereotype, it is the generalisation of three other stereotypes, i.e., the *OperationResource* (UML *Operation* metaclass extension), the *InstanceResource* (UML *Porperty* metaclass extension), and the *ObservableResource* (UML *Operation* metaclass extension). The capability to have multiple instances in the Resource or Object is captured by the stereotypes *ObjectInstance* and *ResourceInstance*.

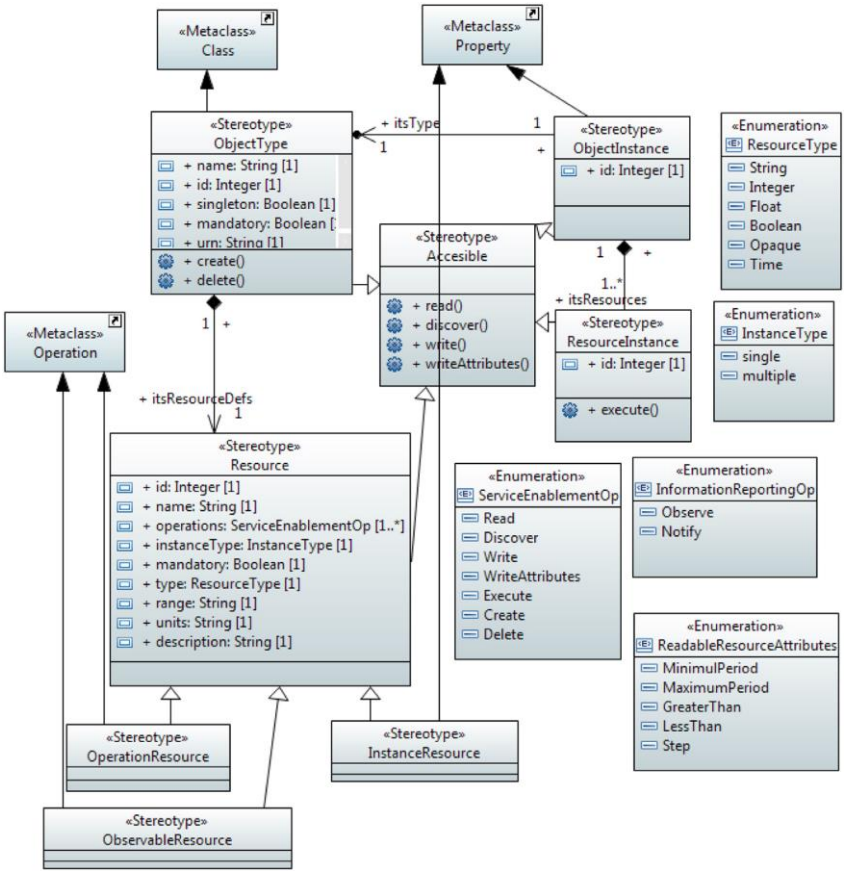

Fig. 2 UML4IoT Profile taken from [22].

For the usability validation the authors used a prototype implementation of the myLiqueur production laboratory system [26], which is, roughly, an IoT system which allows the users to custom

their liqueur, clearly by a remote way using an smartphone app. In Fig. 3 it is shown part of the class diagram from the system cyber part. The interface references as heater attribute are marked with the <<ObjectInstance>> stereotype, the classes that implement their types are marked with the <<ObjectType>> stereotype. On the other hand, all the methods are depicted applying the <<OperationResource>> stereotype.

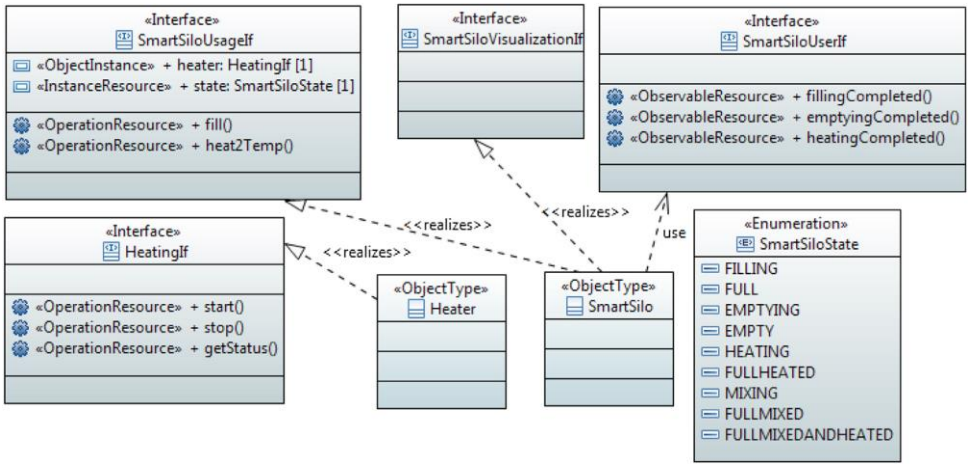

Fig. 3 The cyber interface of the SmartSilo marked with model elements of the UML4IoT profile [22]

In the conclusions of the work the authors mentioned that the adoption of the IoT imposes a paradigm deviation for the automation system developer and it complicates the development process, because of that they proposed UML4IoT. This UML profile eases the conversion process from a LWM2M protocol into an IoT environment. However, it is mainly thought to systems already built and not for systems that are about to be developed. Notwithstanding, as they are using LWM2M protocol, it includes, by its own nature, the very known MQTT and CoAP protocols, which are the leader IoT protocols, this fact makes this profile very useful. On the other hand, and returning to the security requirements in the IoT, as in many systems it continues being an after-thought, since it is not regarded within the UML4IoT profile, thus our proposal would be very easily incorporated within the UML4IoT as it was thought for. The approach proposed in this work is a security layer which allows the developers to design the security requirements within an IoT environment. For our work aim, this proposal may be used or not, it helps the developers to improve their understanding related to IoT and they save time. Hence, the UML4IoT approach would aid the analysis and design stages of an ioT system, but it will be incomplete without a threat analysis and without taking into account the security IoT system security requirements.

2.1.2 IoT-A

IoT-A [27] is a reference model for the IoT domain proposal, developed with the intention of promoting a common understanding of the IoT basis. In this approach a reference architecture is proposed as well, it describes essential building blocks to deal with conflicting requirements considering performance, security, functionality and deployment.

The *IoT Reference Model* proposed is a top-level description which is the highest abstraction level for the definition of the *Architectural Reference Model*. Three models are included inside of this model:

- *IoT Domain Model*
- *IoT Information Model*
- *IoT communication Model*

The *IoT Domain Model* defines relationships between concepts, for instance "Services expose Resources". They provided a common lexicon and taxonomy of the IoT domain. The goal behind the IoT Domain Model is to capture the main notions and their relationships. The generic scenario regarded in the IoT-A proposal is that of a generic User (human or some kind of digital artefact) which has to interact with a *Physical Entity* (PE) in the physical world. The PE has its digital representation that they call *Virtual Entity* (VE) (3D models, data bases, avatars, etc.). The IoT-A approach considers an abstract concept of "thing" such as PE + VE. The relation between VE and PE is usually achieved by embedding VE into, by attaching to, or by simply placing it in close vicinity of the PE. The devices that mediate the relation between the PE and VE are technical artefacts for bridging from the real world to the virtual world, it is done by sensing, actuating, monitoring, computation, storage and processing capabilities. Nonetheless, this definition of device is a bit confusing, since current devices are contemplated as the processing artefact which receive the sensor data or which are able to send data directly to the PE. In an IoT environment the next devices are regarded:

- Sensors: Monitor the PE and they provide information, knowledge, or data about the PE. They can be attached, placed on the environment or embedded into the PE.

- Tags: Are normally attached to the PE, they are used to identify the PE. The identification process is done by a sensor.
- Actuators: They act on the PE, this means that they are able to modify the physical state of the PE.

In the IoT-A the resources are defined as software components that provide data from or are used in the actuation on PEs, and there are identified two resources types: *On-Device Resources* (hosted on devices) and *Network Resources*. A service provides an open interface, where the functionalities are offered, making the resources accessible.

In Fig. 4 the IoT Domain Model is depicted using the UML class diagram. As it can be seen in the figure, hardware is displayed in blue, yellow is used for the animated objects such as humans, animals, etc., green is used for the software elements and the issues that are not clearly classifiable are shown in brown. The User virtually interacts with the PE, therefore, it ("he" is not used since the user can be an electronic artefact) invokes one or more *Services* which is a software and exposes zero or more *Resources* that are a generalisation of *On-Device Resources* and *Network Resources*. The *On-Device Resources* are hosted by the *Device* which is a generalisation of *Tags, Sensors and Actuators*. This approach characterises, in a UML representation, which by its essence is very intuitive, the IoT Domain Model. It helps to clarify the IoT conceptualisation, the relationships between the different agents and although it is not introduced as an UML profile it might be and it would be very convenient, because it would turn it into a metamodel, which would help the specific IoT systems conceptualisation and modelling.

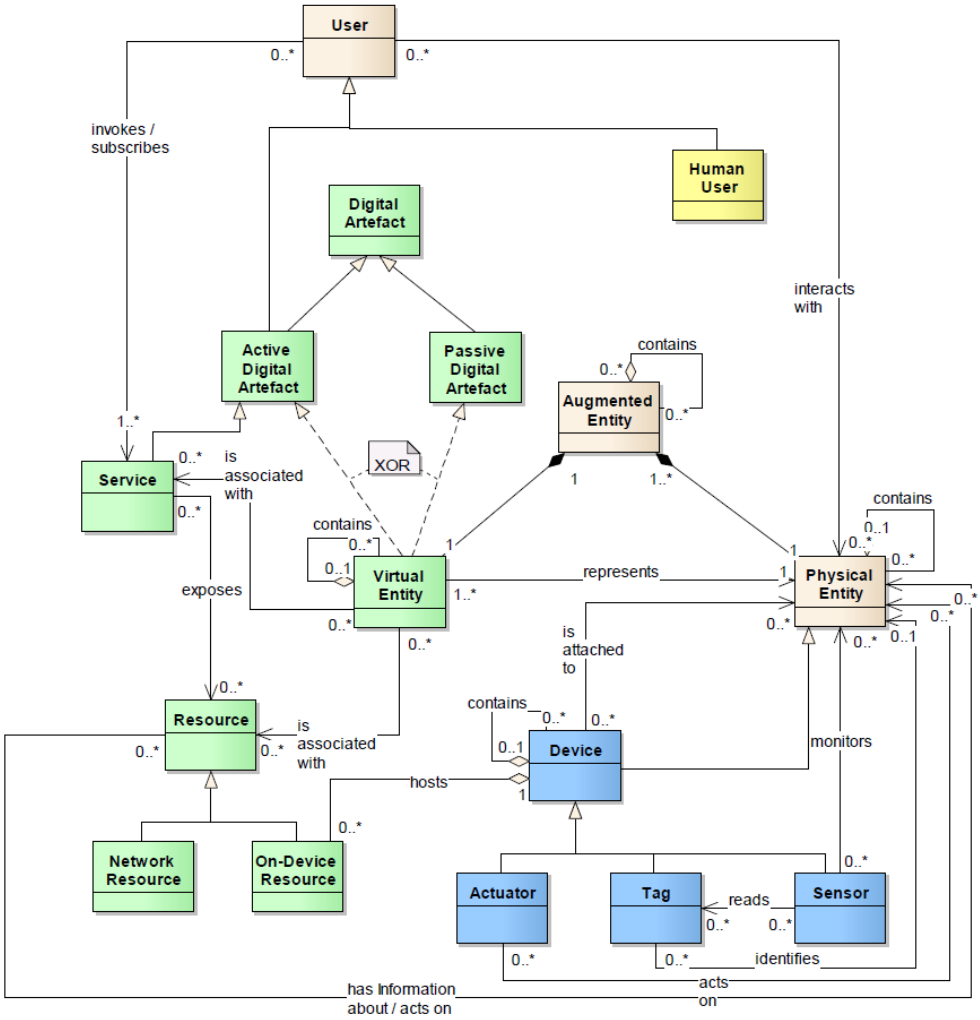

Fig. 4 UML representation of the IoT Domain Model taken from [27]

In IoT-A project the security is regarded, they consider general security modules which are adequate as a first approach to the security issues within the IoT systems, nevertheless, as it was mentioned in the background and will be delver in 2.2.2 section, it does not cover the whole IoT security requirements.

2.1.3 IoTLite

The approach referred to as IoTLite is an instantiation of the semantic sensor network (SSN) ontology [28] to describe key IoT concepts allowing interoperability and discovery of sensory data in heterogeneous IoT platforms by a lightweight semantics in the context of data analytics. In this approach the authors have ten rules for good and scalable semantic model design. They proposed guidelines for developing scalable and reusable semantic models in the IoT. They are much focused on the relatively fast annotation and processing time. The IoTLite proposal can be extended, depending on the applications, different semantic modules can be added to provide additional domain and application specific concepts and relationships.

The guidelines that the IoTLite authors propose for developing scalable ontologies are:

- Design for large-scale.
- Think of who will use the semantics and design for their needs.
- Provide means to update and change the semantic annotations.
- Create tools for validation and interoperability testing.
- Create taxonomies and vocabularies.
- Re-use existing models.
- Link data and descriptions to other existing resources.
- Define rules and/or best practices for providing the values for each property.
- Keep it simple.
- Create affective methods, tools and APIs to handle and process the semantics.

They tried to follow each one of the rules they proposed in their proposal. For instance they created a taxonomy of quantity kinds and units.

The three core concepts proposed in the IoTLite approach are: *iot-lite:object, ssn:Device* and *iot-lite:Service*, these are depicted in Fig. 5. Those concepts are necessary in any ontology describing IoT. In IoT-lite three classes of Devices are considered: *ssn:Sensor, iot-lite:Actuator* and *iot-lite:Tag*, these classes coincide with the IoT-A classification.

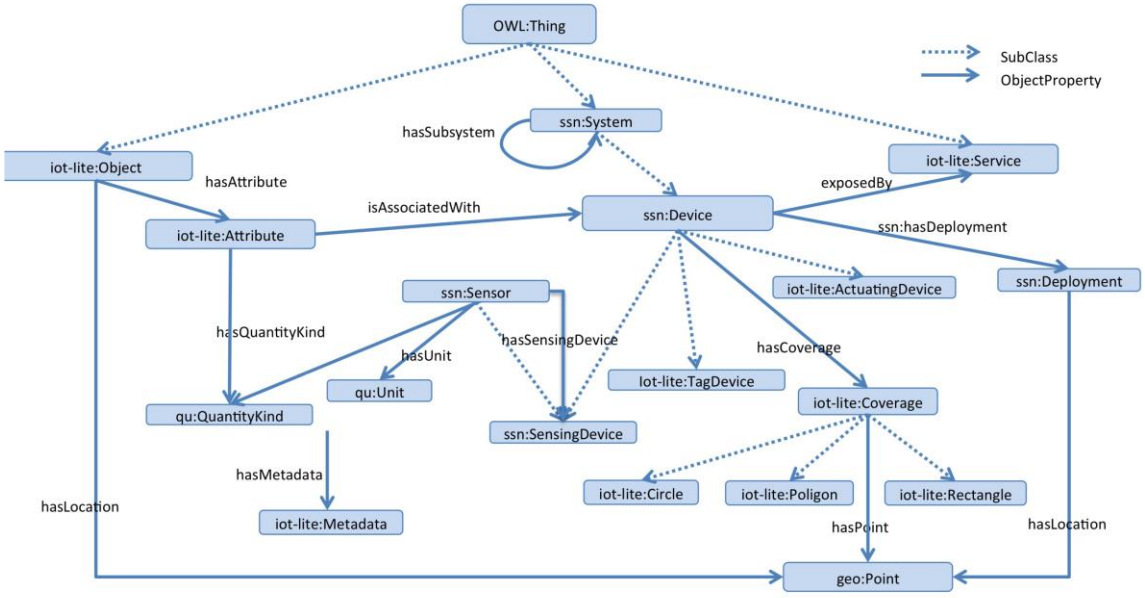

Fig. 5 An overview of the IoT-lite model, taken from [28].

In the IoTLite proposal the security concerns are not considered, it could be because the objective of the proposal is to create a lightweight IoT ontology which is normally opposite to the security concerns addressing. This is because the security mechanisms and countermeasures add processing power, latency and more complex models which will not comply with the rules they proposed. Nevertheless, if the security is crucial for the IoT system then its analysis and targeting may be added to the IoTLite approach.

2.2 UML/SysML Security Extensions

In this section the state of the art about the UML and SysML extensions is introduced. The extensions analysed are: UMLsec, SysMLsec, SecureUML, IoT-A and the IBM Fault Tree Analysis.

2.2.1 UMLsec

The UMLsec approach [17] is a UML extension to model computer system security properties, it is based on a formal semantics. UMLsec aims to support secure system development, in particular through the following goals in an already modelled system (using UML), UMLsec should be able to automatically evaluate it for security-related vulnerabilities in the design.

- The methodology should be available to developers not specialised in security.
- Enable the user to define the security features and properties unambiguously.
- To model security from early design phases.
- Different abstraction levels, since security might be violated on different level.
- UMLsec should be enabled to make use of established rules of prudent security engineering.

In order to target those objectives, the authors followed the next notational features:

- Basis security requirements such as *secrecy* and *integrity* are integrated into their language.
- Consider threat scenarios depending on the adversary strengths.
- Common security concepts.
- Common security mechanisms like access control are included in the notation.
- Cryptographic primitives are defined at an appropriate level of abstraction.
- Physical security
- Security management
- Domain-specific extensions

UMLsec defines precise semantics for security, using UML Machines that allows construction of a single formal description for the system model which includes information from all diagrams and all abstraction layers.

For the UML extension, they focus on the general security properties, those that have a comparatively intuitive and universally applicable formalisation, the aim is to provide the developers with notational elements to depict the most relevant security concerns. In UMLsec *stereotypes*, *tags values* and *constraints* UML extension mechanisms are applied, the core extensions are twenty-one stereotypes which are used to define data security requirements on model elements. The UMLsec stereotypes are shown below, nevertheless there is not a graphic representation within a UML package diagram.

Table 1 UMLsec Stereotypes:

Stereotype	Base Class	Tags	Constraints	Description
fair exchange	subsystem	start, stop, adversary	After start eventually reach stop	Enforce fair exchange
provable	subsystem	action, cert, adversary	Action is non-deniable	Non-repudiation requirement
rbac	subsystem	protected, role, right	Only permitted activities executed	Enforces role-based access control
Internet	link			Internet connection
encrypted	link			Encrypted connection
LAN	link			LAN connection
wire	link			Wire
smart card	node			Smart card node
POS device	node			POS device
issuer links	node			Issuer node
secure links	subsystem	adversary	Dependency security matched by links	Enforces secure communication links
secrecy	dependency			Assume secrecy
integrity	dependency			Assume integrity
high	dependency			High sensitivity
secure dependency	subsystem		<<call>>, <<send>> respect data security	Structural interaction data security
Critical	object, subsystem	secrecy, integrity,		Critical object

		authenticity, high, fresh		
no down-flow	subsystem		Prevents down-flow	Information flow condition
no up-flow	subsystem		Prevents up-flow	Information flow condition
data security	subsystem	adversary, integrity	Provides secrecy, integrity, authenticity, freshness	Basic datasec requirements
Guarded access	subsystem		Guarded objects accessed through guards	Access control using guard objects
guarded	object	guard		Guarded object

As it can be seen there are different abstract levels, for instance the <<secure links>> stereotype refers to the level of abstract syntax whereas <<no down-flow>> refers to the formal semantic. A brief summary of the UMLsec stereotypes is given below, where the main function of each stereotype is described.

<<fair exchange>> stereotype tries to prevent cheating in any transaction, it is a subsystem hence it can be depicted using use case diagrams or activity diagrams (where the tags, {start} and {stop}, are used) applying the stereotype in the subsystem name.

<<provable >> stereotype is a subsystem as well, it has associated tags {actions}, {cert} and {adversary}, it means that given an expression in {cert}, an action in {action} can be performed given an adversary type in {adversary}, the certificate in {cert} is assumed to be unique for each subsystem instance.

<<rbac>> stereotype is a subsystem containing an activity diagram enforces *role-based access control*, thus it is not included any other authentication process such as *attribute-based access control* hence a more abstract authentication stereotyped is required, more information about the <<rbac>> tags can be found in [29].

<<Internet>>, <<encrypted>>, <<LAN>>, <<wire>>, <<smart card>>, <<POS device>> and <<issuer node>> stereotypes are applied on links in deployment diagrams, they denote the communication kind links, although those links stereotypes are very useful when designing the system architecture, they are not completely related to security, since they do not mention for instance *secure communication,* besides they do not cover the whole IoT environment since there are not different layers communication consideration, such as links between sensors, sensor-device, etc. where new technologies are needed (Zigbee, Bluetooth, etc).

<<secure links>> stereotype is used to ensure that security requirements on the communication are met by the physical layer.

<<secrecy>>, <<integrity>> and <<high>> stereotypes may label dependencies in static structure or component diagrams, they depict the respective security requirements. <<secure dependency>> is used for subsystems containing static structure diagrams, this stereotype ensures that for the <<call>> or <<send>> dependency from an object *C* to an object interface *D,* then the message appears in the tag {secrecy} in the object *C* if and only if it does do in *D*.

<<critical>> stereotype labels subsystem instances which contains critical data, using the {secrecy}, {integrity}, {authenticity}, {fresh} and {high} that depict the security requirements. <<no down-flow>>, <<no up_flow>> stereotype enforces secure information flow, it is the UML machine that prevents down-flow (resp. up-flow). <<data security>> stereotype respects the data security requirements given by the stereotype <<critical>>.

<<guarded access>> stereotype means that each object with the stereotype can only be accessed through the objects specified by the tag {guard}. <<guarded>> stereotype labels objects that are supposed to be guarded.

The UMLsec approach uses the UML extension mechanisms to consider security requirements in the model system, besides of providing natural language description and mathematical language definition. Nonetheless, there are some issues about the UMLsec proposal, it is no thought for IoT systems, which makes it incomplete since it does not consider physical tamper protection, new communication protocols, new technologies, etc., and in IoT systems there is a wider attack surface. There are not any new extension within the UML use case diagram which helps the developers to include security requirements in the analysis stage, they give as a fact that the security requirements are known, which is not always the case.

2.2.2 Fault Tree Analysis (FTA), IBM

The IBM FTA proposal [19] is a UML extension to model faults and risks in systems, it was firstly proposed in the year 2010, but it was reintroduce in 2016 with the IoT trend. The IBM FTA proponents argue that UML is a general modelling language that can be extended to model metadata beyond standard usage. Their purpose is to enable upfront analysis of assets, security needs, vulnerabilities, attacks and countermeasures including the ability to link to security-relevant requirements, architectures and design patterns.

The FTA Profile has the following views:

- FTA diagram
- Safety Analysis Diagram (SAD)
- Hazard table view
- Failure Means and Effect Analysis (FMEA) table view
- Fault table view
- Fault- Requirement Matrix view
- Fault – Design Elements Matrix views

UML allows symbols to depict the stereotypes in their extensions, thus in IBM FTA they propose symbols for: <<NormalEvent>>, <<HazardEvent>>, <<BasicFault>>, <<UndevelopedFault>>, <<RequiredCondition>>, <<ResultingCondition>> and <<Hazard>>. All these symbols are shown in Fig. 6. They use logical operators such as: AND, OR, NOT, Transfer Operator, NAND, NOR and XOR, those symbols are depicted in Fig. 6 as well.

The key elements in the IBM FTA metamodel are as follows [30]:

- Hazard: (stereotype of class) it is a condition that will lead to an accident or loss.
- Fault: (stereotype of class) are usually the bottom terminal elements in an FTA and it is the non-conformance of an element to its specification or expectation.
- Resulting condition: (stereotype of class) it is the resulting outcome from a combination of faults and conditions.

- Required condition: (stereotype of class) a condition required for the fault to interact.
- Logical operator: (stereotype of class) it does not have semantics by its own, nonetheless it is applied as a connector.
- Logic flow: (stereotype of flow) it is the connection between a fault, condition or hazard and a logical operator. It can be an input or an output.
- Fault source: (stereotype of class) a normal UML element that could manifest a fault.
- Safety measure: (stereotype of class) a normal UML element that mitigate a fault.
- Manifest relation: (stereotype of dependency) it is a relationship from a fault to a fault source.
- Detect relation: (stereotype of dependency) it is a relation from a fault or hazard to a safety measure which can detect when a fault occurs.
- Extenuates relation: (stereotype of dependency) a relation from a fault or hazard to a safety measure that reduces the likelihood of the hazard or fault.
- Trace to requirement: (stereotype of dependency) it is a relation from a fault or hazard to a requirement.

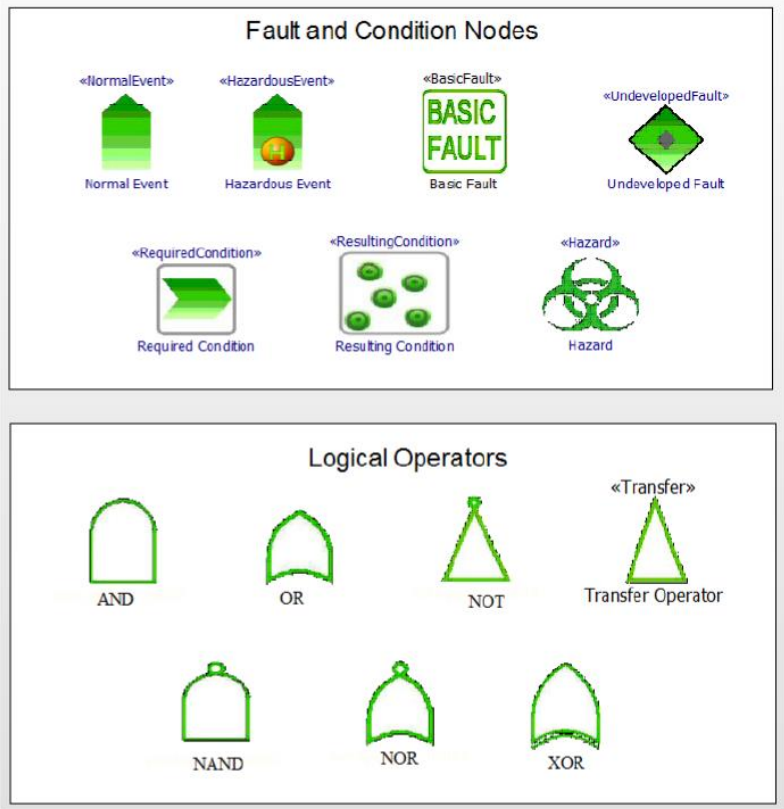

Fig. 6 IBM FTA symbols, taken from [19]

Every diagram proposed within the UML profile has a particular aim, the FTA is a fault tree where the *Hazard* is located as the tree root, and all the possible ways to achieve that *Hazard* or at least the known and logical are depicted as a flow and composition of logical operators. The SAD diagram is in the IBM FTA authors words "like a Fault Tree Analysis (FTA) but for security, rather than for safety", so it tries to find the relation between assets, attack, vulnerabilities and security violations. As the author mentioned, the approach permits reasoning about what kind of security is needed, how many countermeasures are needed, and it allows a risk assessment. Hence, the approach looks very attractive in a first view, it relates countermeasures, threats and vulnerabilities. However, it is a system-independent analysis, thus it does not include system elements within the SAD diagram, which is a fundamental concern for the system conceptualisation and to analyse the system architecture. This is relevant in the IoT domain where many layers can be found also as constrained actors which may not be able to implement a too robust security mechanism. In Fig. 7 there is an example of the SAD diagram where a countermeasure blocks a threat and the <<asset>>, <<security violation>>, <<vulnerability>>, <<resulting condition>>, <<countermeasure>>, <<counteracts>> and <<requiredCondition>> stereotypes are used. The symbols are very intuitive,

nevertheless the blue colour in the <<threat>> symbol is not clear, as well as the red colour in the <<counteracts>>.

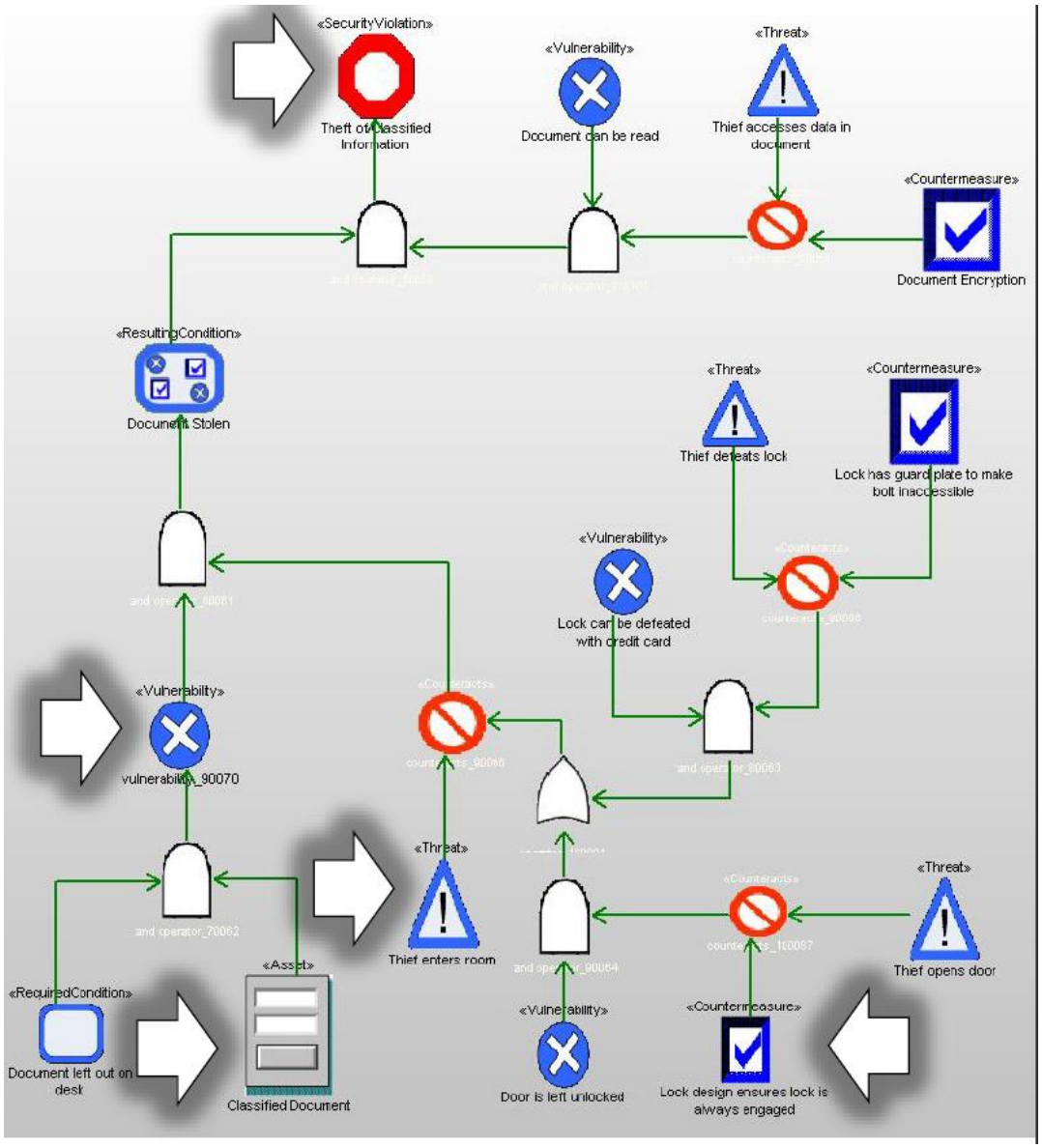

Fig. 7 IBM SAD example, taken from [19].

2.2.3 SecureUML

SecureUML [31] is an UML extension proposed to model RBAC which is a model for access control of users and their privileges are decoupled by roles, it is an authorisation process..

The SecureUML approach is defined as an extension of the UML metamodel (SecureUML metamodel), they depict the RBAC concepts directly as metamodel types. SecureUML introduces new metamodel types, User, Role and Permission, and relations between them, see Fig. 8. Additionally, in their approach the authors introduced the type ResourceSet that depicts a user defined set of model elements. A Permission is a relation object which connects a ModelElement or a ResourceSet to a role and its semantics is defined by the ActionType elements that represent a class of security important operations on a protected resource. The available action types can be defined using ResourceType elements which determine all action types available for a specific metamodel type. The attribute baseClass represents the connection to the metamodel type. An AuthorisationConstraint is a part of the access control policy of an application.

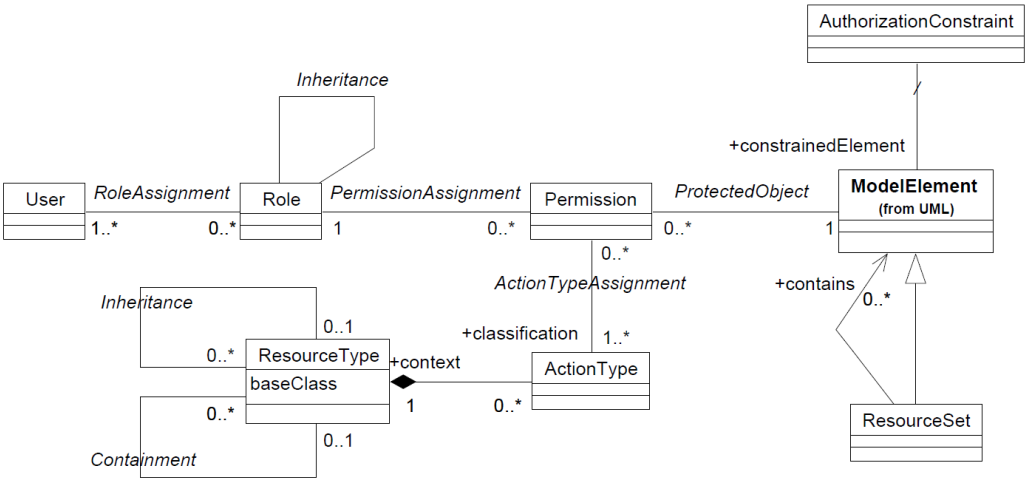

Fig. 8 SecureUML Metamodel, taken from [31].

SecureUML is focused on the access control RBAC, according with [32], although SecureUML only focus on the solution domain and not in the methodological level. The strong feature of SecureUML is the explicit definition of permissions. This approach is applicable for class diagrams,

nevertheless it just covers the RBAC solution, it does not have a threat analysis or something similar and hence it is a UML extension (even when they call themselves *UML dialect*) that is useful to model RBAC but no other security requirements within a system, besides it does not consider IoT systems threats or vulnerabilities.

2.2.4 SysMLsec

SysMLsec [33] is a SysML extension to depict security and safety concerns and attack trees [34] within a SysML environment. It guides and increases the potential for collaboration between system engineers and security experts. Besides, the approach provides detailed representations of the security threats and security requirements. The SysMLsec methodology adopts a three-phase approach: analysis, design and validation, such as a common life-cycle. Nonetheless, this approach regards security concerns along every stage, allowing from the analysis stage, to introduce security countermeasures into the system.

It is not clear the extension process followed by the SysMLsec proponents in order to achieve their approach. Therefore, the system security analysis and design is restricted to the examples they show and the tool they developed (Ttool [35]).

The security requirements are depicted in a SysML requirements diagram using the <<Security Requirement>>, in Fig. 9 there is an example of one diagram.

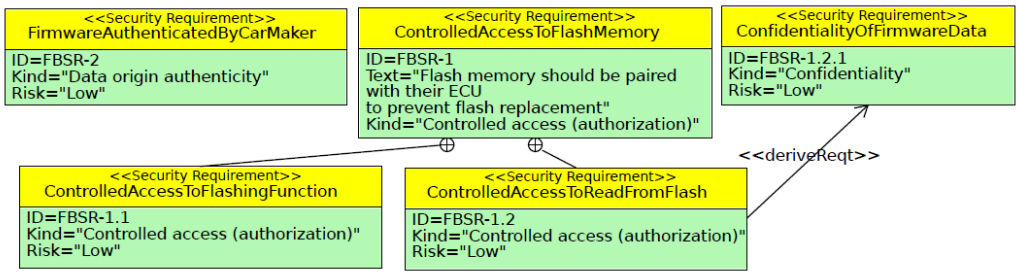

Fig. 9 SysMLSec security requirements example, taken from [33].

The attacks are regarded in a SysML block diagram, it can be deduced that they propose some stereotypes, such as <<AFTER>>, <<OR>> and <<attack>>. The idea of the attack graph is that an attacker tries to violate the system following some steps or subattacks, SysMLSec proposes to order those diagrams sequentially applying the <<AFTER>> stereotype which is something that not appears directly in other approaches such as the presented in [36].

The SysMLsec design is made upon SysML block and state machine diagrams, formally defined in pi-calculus (a process algebra). SysMLsec defines cryptographic blocks ("crypto block") which include a set of cryptographic methods that can be used in the state machines.

2.2.5 IoT-A security model

As it was mentioned in section 1.1.3, the IoT-A proposal did a very large study about the security necessities in the IoT, the approach gathered a set of general requirements [10]. One of the aims of the IoT-A document was to address the security goals which the infrastructure needs to handle. As part of the IoT-A proposal a security model is included, nevertheless the security components they proposed, as it was described earlier, do not cover the IoT universe. Their work is a benchmark for our work, where the security analysis done by them is reapplied in a more reachable and adequate model, extending the UML, as will be introduce in chapter 4.

The security components are the classification results of the analysis done about the IoT security goals (section 1.1.3) and some functionalities where identified. Those functionalities were grouped into five security components. In Table 2 the security components component functionality and the security goals targeted are depicted.

Table 2 IoT-A components [10].

Component name and short name	Component functionality	Security goals targeted
AuthZ (Authorization)	Access control on services	Service access control Confidentiality (data) Integrity (data)

	Access control on resolution infrastructure	Service privacy Service availability
AuthN (Authentication)	Authentication of service users	Authentication Accountability
IM (Identity Management)	Management of Identities, Pseudonyms and related access policies	User privacy Service privacy
KEM (Key Exchange and Management)	Exchange of cryptographic keys	Confidentiality (communication) Integrity (communication)
TRA (Trust & Reputation)	Collecting user reputation scores and calculating service trust levels	Service reputation metering Service trust

As it can be seen from Table 2, the IoT-A proposal applies not just to IoT systems but to informatics systems in a REST architecture. However, the IoT-A approach develops models for each one of the components they proposed, therefore those models need to be in a level down in order to transform them into IoT system architectures. Here is where IoT-A does not comply utterly, since they did not regard any modelling language for the security components. Therefore, it would be hard to instantiate the security components they proposed.

The state of the art approaches include UML extensions for the IoT and security UML extensions for Informatics systems, all those approaches represent the last works related to this proposal, the issues within each proposal were already discussed, besides of identifying the goals to address in IoTsecM, the state of the art helped to see the procedures followed to cover the UML extensions and profile proposition and the security related extensions as well.

3 THEORETICAL FRAMEWORK

3.1 UNIFIED MODELLING LANGUAGE

The notations allow to conceptualise complex ideas, the Real-life projects normally involve many people even around the world with different time zones, languages and culture etc. Therefore, the precision and clarity of the systems notation is fundamental for a good understanding between the individuals. A well-defined semantics allows good communication, so it can be well understood by the project participants.

A model is the abstraction of the features or behaviour of the thing being modelled, it gets the relevant details and depict them in a convenient way for a certain point of view and for a certain purpose. For a system, the model can depict its properties, behaviour or architecture.

The Unified Modelling Language (UML) [37] is a graphical language which is the de facto modelling language in industry, it was obtained according to their proponents: Grady Booch, Ivar Jacobson and Jim Rumbaugh. UML allows modelling, constructing and documenting the software elements of a system, it is mainly used to model object-oriented systems. However, the flexibility of the UML diagrams allows many other domains to be modelled with it. Modelling with UML has several advantages, it is precisely defined, there is a large number of developers trained in UML, and there are many tools to draw the UML diagrams. Hence, UML is a relevant modelling language largely used nowadays.

A UML model has three main categories:

- Classifiers: Describes a set of objects.
- Events: Describes a set of possible occurrences.
- Behaviours: Describes a set of possible executions.

In the UML specification [37] it is stated that UML models do not contain objects, occurrences, or executions of the domain being modelled. On the other hand, UML does have modelling constructs for directly modelling individuals.

The system development focuses on three different systems models:

- Functional model: It is depicted in use case diagrams.
- Object model: Class diagrams, system structure.
- Dynamic model: Sequence diagrams, state diagrams, activity diagrams.

UML diagrams describe different system attributes or behaviour, in the last UML version (2.5) there are many diagrams which will be very briefly introduced below and are classified and displayed in Fig. 10 UML 2.5 diagrams, adapted from [38]. Moreover, the reader can find a lot of information about the diagrams features on the web, books, scientific papers, and in the UML specification.

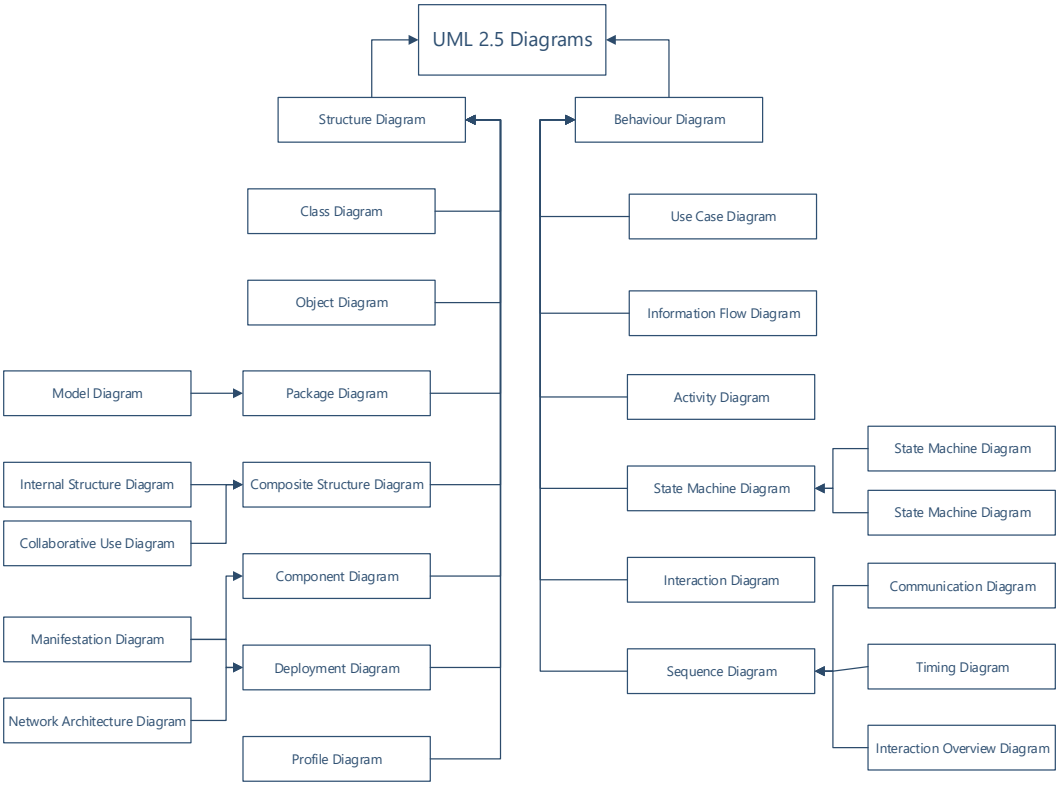

Fig. 10 UML 2.5 diagrams, adapted from [38].

The structure diagrams are:

- Class diagram: It depicts the designed system structure. A class is a description of a set of objects which share attributes, operations, methods, relationships and semantics. The class

diagrams are used to depict the relations, interfaces, features and constraints between the classes.

- Object diagram: An object is class instance, the notation and the interfaces within the object diagram are the same that in the class diagram, just there is a tiny difference, this is that the notation for objects name is object:class. The object diagram is used to depict the instances relations in a graph, it shows a snapshot of the detailed date of the system at a point in time.
- Package diagram: Is a common diagram for all UML diagrams, it groups different packages to depict the system structure.
- Model diagram: Is UML auxiliary structure diagram that displays some abstraction or specific view of a system.
- Composite structure diagram: It depicts the relations between objects, the message sequence and the concurrent execution flows. It displays a class internal structure and its ports. The class meaning in this diagram is wider since it can represent software components as a domain specific class.
- Internal structure diagram: shows the internal structure of the thing being modelled, its parts and their relationships.
- Collaboration use diagram: It shows organised interaction of the system functionality. Normally objects within a system cooperate each other to produce a behaviour of collaboration.
- Component diagram: Components are considered autonomous encapsulated units within a system they are changeable and reusable, they provide one or more interfaces. The component diagram depicts a system temporal understanding which helps the designers. It provides a high-level view, architectonic view, which helps to clarify the system architecture.
- Manifestation diagram: Is used to show implementation of components by artefacts and internal structure of artefacts.
- Deployment diagram: Is applied to depict system physical parts, they can be files, jar or war, the physical does not necessary mean something touchable, like hardware but they can be deployed on hardware. In this diagram the stereotype <<artifact>>is applied.
- Network architecture diagram: UML standard uses Deployment diagrams to allow developers depict network architectures, they do not define specific elements or stereotypes to display the networking concerns.

- Profile diagram: It is a structure diagram that describes extension mechanisms to the UML, in this diagram the stereotypes, constraints and tagged values are defined.

Behaviour diagrams

- Use case diagram: It can be used to depict interaction between the designed system use cases and a user in an abstract way. The main elements are use cases (a use case is a sequence interaction produced when an actor interacts with the system), actors and their relations (stereotyped <<include>>, <<extend>> and <<generalize>>).
- Information flow diagram: It depicts information flow between entities at high level abstraction, it is mainly applied when a clarification about some use case is needed. It is useful to describe system information flow.
- Activity diagram: Is an especial case of a state machine diagram, where one or more objects behaviour and conditions are shown, the execution is coordinated by the activity diagram. They are commonly called control flow and object flow models. They are used to model the system dynamics aspects and describes the state changes an object experiments.
- State Machine diagram: It depicts the finite states along the object life, it helps to clarify complex objects activities, since inside of the object there are some activities which are displayed in the state machine diagram. The objects have behaviour, they do things and they know things.
- Behavioural state machine diagram: Is a specialisation of behaviour used to specify discrete behaviour of a part of designed system through finite state transitions.
- Protocol state machine diagram: Is a state machine diagram applied to specify protocols or a lifecycle of some classifier.
- Interaction diagram: It includes sequence diagrams, communication diagrams, timing diagrams and interaction overview diagrams.
- Sequence diagram: It shows the objects interactions in time sequence, it shows the sequence of messages exchanged, there are two kinds of sequence diagrams, the first one is a generic form where all possible scenarios are describes and the other is in an instance form where one current scenario is described. The sequence diagrams depict the participant objects and their message ordered interaction.
- Communication diagram: Is a kind of UML interaction diagram, which shows interactions between objects applying sequences messages in a free-form arrangement.

- Timing diagram: It helps to reasoning about the models on time situations, what is very important about time constraints.
- Interaction overview diagram: It provides an overview of the flow of control of interactions. This diagram combines elements from the activity and interaction diagrams, it is easily replaced with times diagrams or sequence diagram.

3.2 UML EXTENSION

3.2.1 UML profile

When the UML elements syntax or semantics cannot express specific concepts of particular systems or domains, the profiles can be applied to define some constraints added to the UML notation with the aim of defining extensions to UML that allows the modelling of specific systems. The Object Management Group (OMG) defines two possible approaches for defining domain specific languages [39]:

- A new language, alternatively to UML or any other existing language. In this case both, the syntax and semantics are defined to fit the specific system or domain features. This is achieved following the Meta Object Facility (MOF).
- An extension to UML where the original UML notation is not modified, nonetheless new constraints are added regarding the UML and OMG extension rules.

The profile describes capabilities that allow metaclasses to be extended in order to adapt them for different purposes. It is the lightweight extension mechanism provided by UML which allows the extension and specialisation of the UML meta-model.

When defining a UML profile, its stereotypes are defined to extend the UML classes in the normative version of the UML metamodel [37]. The profiles are straightforward mechanisms for adapting an existing metamodel with constructs that are specific to a particular domain. Thus, a UML profile allows to add new constraints (domain-specific) to the UML metamodel.

A model is a description of (part of) a system written in a well-defined language. A well-defined language is a language with well-defined form (syntax) and meaning (semantics) [39].

The OMG defines a four-layered architecture that separates the different abstraction levels making up a model [39]:

- Layer M0: Instances. The M0 layer models the running system and its elements are the software implementation, hardware or any element that interacts or is part of the systems, it is the actual instances that exist in the system.
- Layer M1: The model of the system. The elements of the M1 layer are models, such as a UML system model. This layer describes the classes each one with associated attributes. The elements of the M1 layer are classifications of elements in the M0 layer. Each M0 element is an M1 element instance.
- Layer M2: The model of the model (the metamodel). The modelling languages are placed at this level such as UML. Layer M2 describes the elements used to model an element in M1. In the case of UML, layer M2 defines “Class”, “Attribute”, “Association”, etc. Every element at M1 is an instance of M2 and every element at M2 categorizes the M1 elements. The model that resides at the M2 layer is called metamodel.
- Layer M3: (the meta-metamodel), M3 describes the concepts to model a modelling language. The concept of UML Class is an instance of the corresponding element of M3 that defines what a class is and its relationships with the rest of UML concepts. The modelling language to describe the M3 elements is MOF and MOF can be defined by itself.

The UML specification [37] describes seven principal reasons why UML needs to be extended:

- Give a terminology that is adapted to a particular platform or domain.
- Give a syntax for constructs that do not have a notation.
- Give a different notation for already existing symbols.
- Add additional semantics to UML specific metaclasses.
- Add types that do not exist in UML.
- Add constraints that restrict the way UML’s constructs are used.
- Add information that can be used when transform a model to another model or code.

The UML metamodel provides three extension mechanisms which allows to customise the UML to specific domains, these three mechanisms are:

- The stereotype is the principal extension mechanism provided by UML, it is a text between angle brackets (chevrons) that is added as metamodel elements, in order to add semantics to

the UML metamodel. A stereotype is defined by a name and by a set of metamodel elements it can be attached to. Graphically, they are defined within boxes, stereotyped <<stereotype>>, the metamodel elements are stereotyped <<metaclass>>. The notation for an extension is an arrow pointing from a stereotype to the extended class. If the stereotype extends to more than one metaclass, then it can only be applied to exactly one instance of one of those metaclasses at any point of time. Any UML metaclass can be extended using stereotypes such as States, Transitions, Activities, Use Cases, Components, Properties, Dependencies, etc., and each metaclass can be extended by as many stereotypes as it is required, it applies as well in backwards, this is one stereotype is able to extend one or more metaclasses.

- Constraints are a set of well-formed rules detailed in Object Constraint Language (OCL) or in natural language and it is the way of describing the semantic [40], for instance in Table 1 the constraints are described in natural language. The constraints can be associated to stereotypes, imposing restrictions. Some examples of constraints are pre- and post-conditions of operations, invariants, derivation rules for attributes and associations, etc. [39].
- The tagged value is and additional meta-attribute, it is attached to a metamodel metaclass. Tagged values have *name* and *type* besides they are associated to a stereotype. They are depicted as graphicall attributes of the class defined as stereotype.

Images can also be displayed, the Image class provides the necessary information to display an Image in a diagram. They are attached to stereotypes and they can be used in lieu of, or in addition to. Some model elements already use an icon, for instance the actor's icon (stickman).

3.3 SysML

SysML [41] is a visual modelling language which provides semantics and notation, SysML resuses a subset of UML 2 diagrams and notation and extends other parts. The intersection between UML and SysML is named UML4SysML and the extensions to UML is named SysML profile.

SysML was designed to provide the developers a profile which was able to describe requirements within diagrams, structure, behaviour, allocations and constraints on system properties in order to support engineering analysis. In Fig. 11 the SysML diagrams taxonomy is introduced, the diagrams used from UML and do not change are sequence diagram, state machine

diagram, use case diagram and package diagram. They modify three UML diagrams, the activity diagram, block definition diagram and internal block diagram. Finally, SysML proposes two new diagrams which are requirement diagram and parametric diagram.

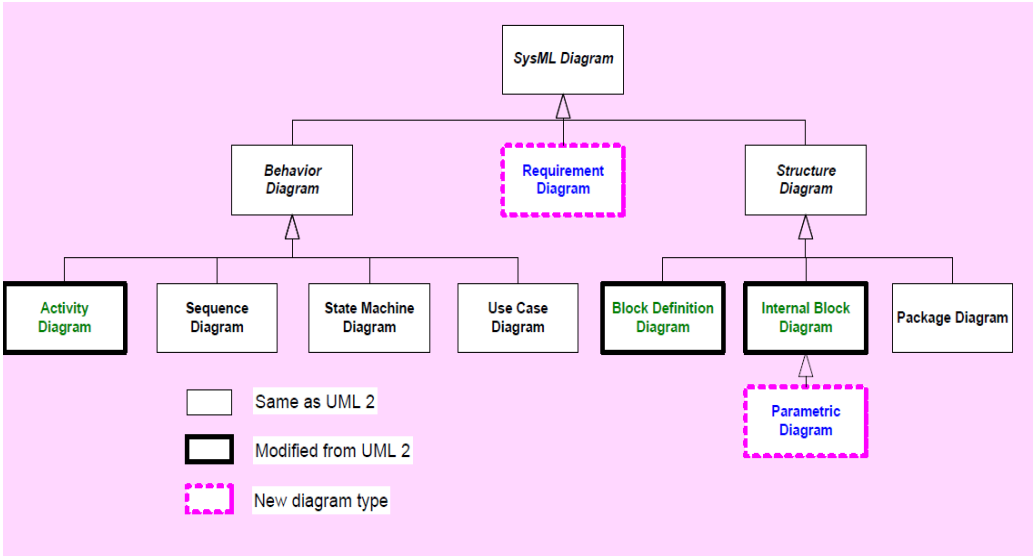

Fig. 11 SysML Diagram Taxonomy, adapted from [42].

One of the most important extensions provided by SysML is the conception of 'block', which provides a unifying idea to detail the structure of an element such as system, hardware, software, data, procedures, facilities and even person. The <<block>> stereotype is an extension of a UML class from UML composite structure; therefore, its notation is very similar, this means that the <<block>> stereotype is located in the first compartment up in the box followed by the block name. Then, multiple standard compartments can describe the block characteristics such as properties, operations, constraints, allocations from/to other model elements and requirements that the block satisfies.

The modified and new diagrams types will be briefly explained in this part, on the other hand, the UML diagrams that were not changed were explained in section 3.2:

- Activity diagram: Activities specifies inputs conversion to outputs through a controlled sequence of actions. The extensions proposed for the activities are: support for continuous flow modelling and alignment of activities with enhanced functional flow block.
- Block Definition Diagram (BDD): Describes the relationships among blocks (e.g. composition, association, specialisation).

- Internal Block Diagram (IBD): Describes the internal structure of a block in terms of its properties and connectors.
- Requirement Diagram: The <<requirement>> stereotype represents a text based requirement. The requirements have hierarchy and relations such as DeriveReqt, Satisfy, Verify, Refine, Trace and Copy.
- Parametric Diagram: It is additional to the IBD and includes all its notation, nevertheless it is restricted and firstly all the constraint blocks must be labelled with “parameters” that contain declaration of the parameters.

The SysML basis was already introduced were the new diagrams were very briefly described, nonetheless we encourage the reader to inquire more into the notation and rules, these can be found in [41]. At this point it is possible to introduce the extension mechanisms of SysML. SysML specification uses the next extension mechanisms in order to define SysML extension as we can witness the extension mechanisms are the same that in UML:

- UML stereotypes
- UML diagrams extensions
- Model libraries

SysML diagram extensions define new diagram notations that supplement diagram notations reused from UML 2. SysML model libraries describe specialised model elements that are available for reuse [41].

As in UML the Profile package contains mechanisms that extend metaclasses from existing metamodels. Therefore, the profiles mechanism is consistent with the OMG MOF. The new notational extensions added by SysML depict stereotypes properties in compartments as well as notes.

The requirement extension is applying stereotypes as well, thus the requirements may be extended to create <<functionalRequirement>>, which would allow specific properties and constraints.

3.4 VULNERABILITIES, THREATS, RISKS AND ATTACKS FOR IOT

The IoT security was already introduced in chapter 1, in this section a brief introduction of the threats, vulnerabilities and risks will be given as security requirements which will be classified.

First, it is important to delineate the differences between threats and risks. A threat is the exploit potential, for instance in the case of a car burglar, the first thought might be that the burglar is actually the threat, nevertheless he is the attack source who is normally motivated by his selfish feeling of wanting something that does not belong to him, in this context the threat is the potential for the burglary to be performed [7]. The threats in the IoT appear in each system layer, they consist of threats against the data assurance, communication assurances, application assurance, etc.. Some examples of IoT threats include physical security, hardware, software quality, environmental, supply chain, CPS physical reliability, resilience threats, control system transfer functions, state estimation filters (kalman filters), insider threats (when individuals within an organisation misuse their privileged access to cause a negative impact on the confidentiality, integrity or availability of the organisation systems) [43]. This is not just for organisations, it could be, for instance, a sensor network inside of a house, the people who has access to the house such as a plumber may easily plug a malicious sensor, or device. Another threat is for instance the data deluge caused by billions of entities generating information, this is a privacy threat. The IoT threats analysis can be addressed from different angles, for instance in [44] an analysis in a 4 layer architecture (end-node, sensing layer, network layer and service layer) was done. Another kind of threats are the ones that directly affect the human user, for instance the tracking and profiling user. In [45] a set of threats regarding the IoT scenario is introduced here as well:

- Control systems, vehicles and even the human body can be accessed and manipulated causing injury or worse: Unauthorised access, actuation and control systems.
- Health care providers can improperly diagnose and treat patients.
- Loss vehicle control
- Safety-critical information unnoticed such as warnings of a broken gas.
- Critical infrastructure damage
- Malicious parties
- Unanticipated leakage of personal or sensitive information
- Unlawful surveillance

- Inappropriate profiles and categorisations
- Manipulation of financial transactions
- Monetary loss
- Theft or destruction of IoT assets
- Gain unauthorised access to IoT edge devices.
- Unauthorised access to the enterprise network
- Create botnets, compromising many IoT devices.
- Impersonate IoT devices.

Once the most general IoT threats were introduced is time to think about the system weakness. Vulnerabilities are unavoidable in all systems, there are design vulnerabilities, implementation vulnerabilities and protocol vulnerabilities, they can be deficiencies in a device physical protection, software quality, configuration, suitability of protocol security, deficiencies in hardware, internal physical architecture and interfaces, operating system and the list goes on, in fact there are new vulnerabilities every day. A vulnerability is the goal of the current exploit from the threat actor.

The Open Web Application Security Project (OWASP) lists the top ten IoT vulnerabilities [46]:

- Insecure web interface
- Insufficient authentication/authorisation
- Insecure network services
- Lack of transport encryption/integrity verification
- Privacy concerns
- Insecure cloud interface
- Insecure mobile interface
- Insufficient security configurability
- Insecure software firmware
- Poor physical security

The same project (OWASP) [46] lists the vulnerabilities related to the attack surfaces they identified.

Table 3 OWASP IoT attack surfaces and vulnerabilities

Attack Surface	Vulnerability
Ecosystem Access Control	○ Implicit trust between components ○ Enrolment security ○ Decommissioning system ○ Lost access procedures
Device Memory	○ Cleartext usernames ○ Cleartext passwords ○ Third-party credentials ○ Encryption keys
Device Physical Interfaces	○ Firmware extraction ○ User Command line interface (CLI) ○ Admin CLI ○ Privilege escalation ○ Reset to insecure state ○ Removal of storage media
Device Web Interface	○ SQL injection ○ Cross-site scripting ○ Cross-site Request Forgery ○ Username enumeration ○ Weak passwords ○ Account lockout ○ Known default credentials
Device Firmware	○ Hardcoded credentials ○ Sensitive information disclosure ○ Sensitive URL disclosure ○ Encryption keys ○ Firmware version display and/or last update date
Device Network Services	○ Information disclosure ○ User CLI

	○ Administrative CLI ○ Injection ○ Denial of Service ○ Unencrypted Services ○ Poorly implemented encryption ○ Test/Development Services ○ Buffer Overflow ○ Vulnerable UDP Services ○ DoS
Administrative Interface	○ SQL injection ○ Cross-site scripting ○ Cross-site Request Forgery ○ Username enumeration ○ Weak passwords ○ Account lockout ○ Known default credentials ○ Security/encryption options ○ Logging options ○ Two-factor authentication ○ Inability to wipe device
Local Data Storage	○ Unencrypted data ○ Data encrypted with discovered keys ○ Lack of data integrity checks
Cloud Web Interface	○ SQL injection ○ Cross-site scripting ○ Cross-site Request Forgery ○ Username enumeration ○ Weak passwords ○ Account lockout ○ Known default credentials ○ Transport encryption

	○ Insecure password recovery mechanism ○ Two-factor authentication
Third-party Backend APIs	○ Unencrypted Personally Identifiable Information (PII) sent ○ Encrypted PII sent ○ Device information leaked ○ Location leaked
Update Mechanism	○ Update sent without encryption ○ Updates not signed ○ Update location writable ○ Update verification ○ Malicious update ○ Missing update mechanism ○ No manual update mechanism
Mobile Application	○ Implicitly trusted by device or cloud ○ Username enumeration ○ Account lockout ○ Known default credentials ○ Weak passwords ○ Insecure data storage ○ Transport encryption ○ Insecure password recovery mechanism ○ Two-factor authentication
Vendor Backend APIs	○ Inherent trust of cloud or mobile application ○ Weak authentication ○ Weak access controls ○ Injection attacks
Ecosystem Communication	○ Health checks ○ Heartbeats ○ Ecosystem commands ○ Deprovisioning

	○ Pushing updates
Network Traffic	○ LAN ○ LAN to Internet ○ Short range ○ Non-standard

On the other hand, the risks depend on the probability of a particular event, attack, or condition. It can be measured applying qualitative or quantitative methods. It is different of the vulnerabilities since the vulnerabilities are the exploitable spot. On the other hand, the risk is the impact that an event can cause measured in money or quality of service for instance. It depends a lot on the attacker possibilities such as knowledge, money, number of attackers, expertise of the attacker, etc., an attack model is useful to model the attacker characteristics. Risk is the damage that a motivated attacker can injure into the system.

Threat modelling helps to manage the risk, because once the threats are known, normally, we are able to rate them in order to balance the risks that the system is ready to affront, the threat modelling helps to measure the impact and overall cost of a compromise, how valuable the target may be to attackers, anticipated skill and motivations and a priori knowledge of a systems vulnerabilities and their repercussion.

The security requirements for IoT are completely related to the risks, vulnerabilities and threats, IoT security requirements depend on the system, since every system has independent assets to be protected, therefore the general vulnerabilities may not apply to every system, despite of that the security requirements must intent to avoid the vulnerabilities and threats and they must reduce the risks, hence a customised analysed should be done for each IoT systems, or even for the future IoT devices development. The IoT should provide applications with strong security protection.

The security requirements are the vulnerabilities and threats and attacks but with the “not allow” (or any other negation in order to prevent the attack, threat or vulnerability) at the beginning for instance:

- Threat: Injection attack
- Security requirement: Not allow injection attack.

There are different abstraction levels for the security requirements, the example above is a low-level abstraction security requirement, nevertheless those can be mapped to a more general requirement, for instance authentication, confidentiality, availability, non-repudiation, etc.

In a four layer IoT architecture, in [44] a security analysis was performed, it is important to point at the security requirements for each IoT layer, therefore, the security requirements per layer will be described here based on [44]:

- Sensing layer and IoT end-nodes:
  - End nodes: Physically security protection, access control, authentication, non-repudiation, confidentiality, integrity, availability, privacy, etc.
  - Sensing layer: data source authentication, device authentication, integrity, availability, timeless, etc.
- Network layer:
  - Confidentiality, integrity, privacy, protection, authentication, group authentication, keys protection, availability, etc.
  - Privacy leakage, since IoT devices may be located in untrusted places, an attacker can retrieve the confidential information.
  - Communication security (Integrity, confidentiality and availability).
  - Allow over connectivity to do not lose control of the user, nevertheless here a DoS attack may appear; therefore, the existence of a behaviour monitor is needed.
  - Protection against man in the middle (MITM) attack.
- Service layer:
  - Authentication, service authentication, group authentication, privacy protection, integrity, security of keys, non-repudiation, anti-replay, availability, etc.
  - Privacy leakage.
  - Application-interface layer:
  - Integrity and confidentiality, cross layer authentication and authorisation.

In this chapter the theoretical basis were introduced, this means that all the fundamental knowledge and works which the research includes were explained in order to be able to propose the UML/SysML extension to depict and model the IoT security requirements. In the next chapter

the IoTsecM proposal is formally presented, applying the UML/SysML extension mechanisms described in Chapter 3.

4 IoTsecM: Methodology and research development

In this chapter the proposal extension approach to UML/SysML is introduced. This proposal is referred to as IoTsecM (IoT Security Modelling) since it is aimed at modelling the security requirements of IoT systems. Firstly, the security requirements were obtained and analysed in the previous sections, from there, fourteen security elements were identified, which are abstract enough to depict the security concerns of IoT systems from the analysis stage in a MBSE process, this allows the developers to add security mechanisms in forward stages such as design, even if the developers are not completely related to cybersecurity.

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

The UML/SysML extension is the way chosen to deploy the proposed security nomenclature in an intuitive and graphic notation, which escorts the developers along the analysis and design stage, it is meant to decide where to introduce security countermeasures in the IoT system.

IoTsecM has two main attributions:

- IoTsecM actors: this is introduced in section 4.1, they model the principal actors in an IoT environment (humans, actuators, sensors, tags and IoT devices).
- IoTsecM nomenclature: It comprises fourteen security elements, it is the IoTsecM core and it will be introduced in section 4.2.

The profile design development starts once the IoTsecM metamodel was designed, the security nomenclature and actors were obtained, therefore it has to fit into a UML/SysML profile, in order to achieve that the metamodel was deployed on eclipse Papyrus a tool which allows the profile generation. There, the stereotypes, constraints and tags were defined. Although in the next sections the IoTsecM profile will be explained in detail, an IoTsecM profile nomenclature overview is displayed in Fig. 12 and the IoTsecM profile actors overview is shown in Fig. 13. The IoTsecM profile applies the UML and SysML metamodels, in order to obtain the metaclasses features from each one.

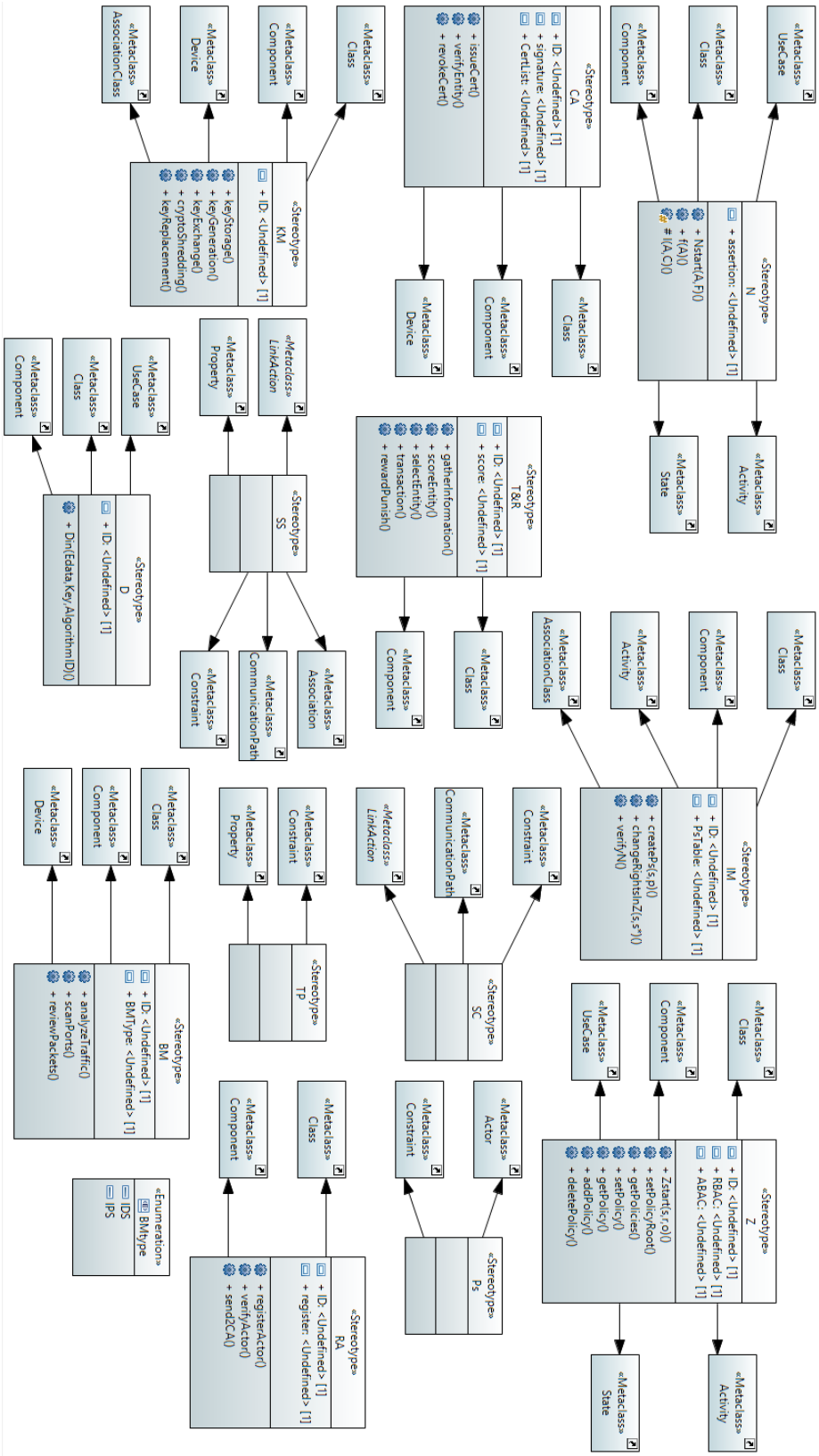

Fig. 12 IoTsecM profile nomenclature overview

4.1 IoTsecM actors

Within the IoT landscape, the interaction of many entities may appear according to the IoT system. Therefore, the IoT environment comprises many assets from many domains, this last fact makes the IoT an immense compendium of items. There are some approaches that help in the actor's classification such as IoT-A proposal specifically in their domain model, where they identified users and devices.

The proposal of actors does not rely on the security concerns directly, nevertheless they are a fundamental part of the modelling stages since they abstract the main features of each actor, allowing the modelling process. In IoTsecM there are four actors regarded as:

- User: The 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: It is any artefact that provides information about the physical entity.
- Actuator: It is any artefact that is able to modify the physical state of a physical entity.
- Tag: It is normally attached to the physical entity and it allows its identification.
- IoT device: It is the hub and processing core which gathers the sensor information and processes it. The IoT handles the communication from the virtual entity, to the system. It is able to send instructions to actuators.

The UML extension mechanisms applied for the actors are stereotypes and the UML metaclass extension is Actor, it is part of the IoTsecM profile and the corresponding part is depicted in Fig. 13.

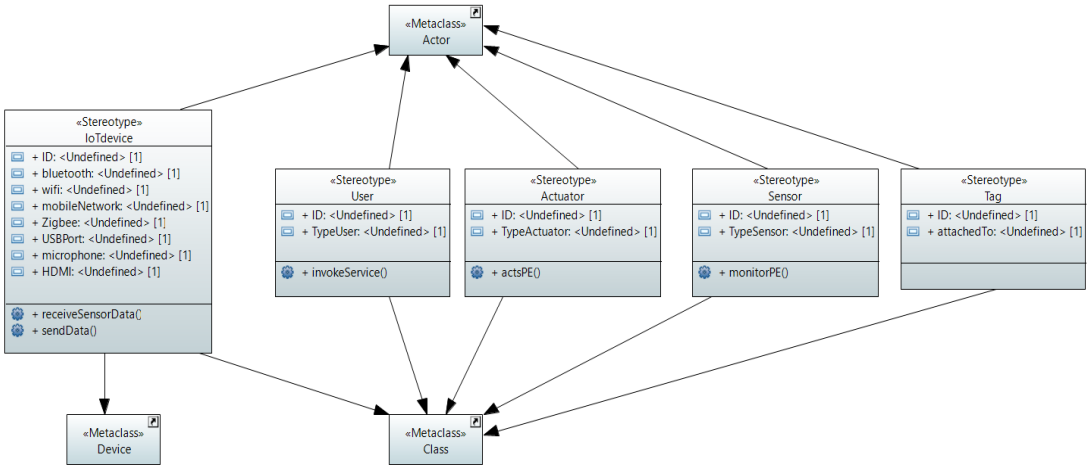

Fig. 13 IoTsecM profile actors.

The IoTsecM stereotypes to depict the 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 introduced below.

IoTdevice

The IoT device extends the UML metaclasses *Device* (from the deploy diagram) because it can be used to model the system architecture as a hardware device, <<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. The IoT device has some predefined attributes which abstract the main communication capabilities such as:

- *Bluetooth:* This attribute is thought to be Boolean type, nevertheless it was predefined as Undefined to allow the developers flexibility when they apply the profile. When it is true then the Bluetooth communication is activated, when it is not it means than that capability is not supported, or if it is supported it is not active.
- *Wifi:* It is recommendable to define this attribute as Boolean type, its semantic is quite similar to the *Bluetooth* attribute and nevertheless this attribute is applied to Wi-Fi connection.
- *MobileNetwork:* This attribute is a Boolean type as well, it defines when the mobile network communication is activated or not.

- *Zigbee:* It is a very common communication protocol in wireless sensor networks (WSN), as the IoT device gathers the sensor data or sends instructions to the actuators, therefore, it is relevant to add this protocol as a predefined attribute in boolean type in order to know when it is enabled or disabled.
- *USBPort:* Many IoT devices have USB ports, hence they are regarded as an IoT device attribute. It is recommended that *USBPort* attribute be defined as Boolean as well.
- *Microphone:* If the IoT devices owns a microphone it can be depicted as well with this attribute defined as Boolean type.
- *HDMI:* It depicts a High Definition Multimedia Interface (HDMI) port which is shown as a Boolean attribute as well.

Two operations are regarded as predefined in <<IoTdevice>> stereotype in order to depict the most common functionalities of this module which are:

- *receiveSensorData()*: This operation is meant to gather the sensor data in the IoT device.
- *sendData()*: This is a more general operation, it is able to send data to the user if the corresponding service is running on the IoT device. Another option is to send information to some external server or the cloud.

User

The <<User>> stereotype is applied to display the existence of a user, as it was mentioned earlier it 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 might be applied as an actor in a use case diagram and as a class in a class diagram, or in an object diagram when instance(s) of it appears.

The attributes proposed for the <<User>> stereotype are ID which is unique and is recommendable to state it as Integer, nevertheless in the stereotype it is undefined in order not to be 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.

Actuator, Sensor and Tag

The <<Actuator>> stereotype refers to the entity which is able to 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 to interact directly with the physical entity.

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 sensor type is defined, for instance 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.

The <<Tag>> stereotype models the tag actor, it includes a unique *ID* attribute for each tag and it defines the *attachedTo* attribute which indicates which physical entity it is attached to.

4.2 NOMENCLATURE

The aim of the IoTsecM profile is to address the security concerns of IoT systems, therefore, the core of the proposal are the security elements which are described in this section. They encapsulate the security requirements in security elements besides each element conforms a nomenclature which is the way to depict them in a UML/SysML profile since it is easier to memorize, reduce boxes sizes and allows a more agile design.

The number of elements identified is fourteen, each one is extended applying a UML/SysML extension mechanism, a summary of these elements is depicted in Table 4 and in the next subsection a detailed description is introduced.

Table 4 IoTsecM profile nomenclature.

Element	Name	Extension mechanism	Base meta-class(es)
N	Authentication	Stereotype	Class, use case, component, block, activity and state

Z	Authorization	Stereotype	Class, activity, component, block, state and use case
C	Cypher	Stereotype	Use case, component, block, class
D	Decipher	Stereotype	Use case, class and component
SS	Secure Storage	Stereotype	Link, property, association, communication path and constraint
SC	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
IM	Identity management	Stereotype	Class, block, component, activity and association class
Ps	Pseudonym	Stereotype	Actor and constraint
CA	Certification authority	Stereotype	Class, block, component and device
RA	Registration authority	Stereotype	Class, block and component
TP	Tamper protection	Stereotype	Constraint and property
BM	Behaviour monitor	Stereotype	Class, block, component and device

In next subsections each one of the IoTsecM nomenclature elements are described.

4.2.1 Authentication: N

Authentication is the security mechanism for ensuring that the identity of a user or service is valid, hence it is an essential element of a typical security model. The term authentication defines the process of verifying the identity of a subject to prove if someone or something is, indeed, who or what it claims to be. “Authentication is the binding of identity to a subject” [47].

In IoT systems the authentication process is not just in one layer of the system, it can be spread along the whole architecture, for instance the physic layer must authenticate sensors, tags and actuators, or the service layer has to validate the IoT devices identity. Therefore, in IoT systems the authentication mechanism becomes more complex, hence, it needs to be part of an entire security infrastructure for its correct implementation.

The authentication regards three main types [48]:

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

A robust authentication process should comprise the binding of two or more authentication types. The functional principle is addressed with a formal description which is provided by [47] and it consists of five components:

1. A: Set of specific information with which entities prove their identities.
2. C: Complementary information, it is the set of information that the system stores and uses to validate the authentication information.
3. F: Complementation functions that generate the complementary information from the authentication information. That is, for $f \in F$, $f: A \rightarrow C$.
4. L: Set of authentication functions that verify identity. That is, for $l \in L$, $l: A \times C \rightarrow \{true, false\}$.
5. S: Set of selection functions that enable an entity to create or alter the authentication and complementary information.

The authentication process consists of obtaining the authentication information from an entity, analysing the 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 managing the data are required.

When the IoT security architecture is defined the authentication, capability is one of the most important security technology selection and it will depend on the deployment designs for the IoT infrastructure e.g., if the Amazon Web Services (AWS) IoT cloud is used then the authentication and

authorisation built-in should be examined. Nowadays Amazon provides two options: X.509 certificates and Amazon own SigV4 authentication, they offer two protocol choices: MQTT and HTTP. Nevertheless, if the IoT system is thought not to use any cloud-based service it may leverage a public key infrastructure (PKI) certificates for authentication or a pseudonyms public key infrastructure (PPKI) as well [49]. The industry has noticed that using secure socket layer (SSL) certificates are not practical, hence they have tried some other vendors such as GlobalSign and DigiCert. Other vendors that offer IoT-specific authentication and authorisation solutions are Brivo, ForgeRock and Nexus.

The element N abstracts the authentication process in a model item guaranteeing for the occurrence of an authentication of an actor applying an authentication method at a particular time, N is applied in the IoTsecM diagrams as a stereotype. Once the security requirements and non-functional requirements are obtained the N element depicts an authentication mechanism regarding the abstraction level, this means that it can model the security requirement, use case, class, object, component and block. Therefore, the metaclasses extended are class, object and component and block as it is shown in Fig. 16. This certainly helps IoT designers to build security mechanisms from a design stage even if they are not completely involved in security issues.

For the IoTsecM actors, the N element is applied as a security requirement and it can be expressed with an N over the actor's head, this is an extension to the UML notation since UML does not depict security requirements in the use case diagram, as is the case also in SysML and SysMLsec. This will allow the designers to build the first use cases from the scenarios found, addressing the authenticated actors properly in a visual way. In Fig. 14 the authenticated actors are depicted, where the N is written within a text box since it can be implemented in all UML/SysML tools.

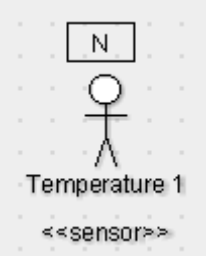

Fig. 14 <<N>> stereotype as a requirement over the actor's head.

When the authentication process is considered as a use case, the <<N>> stereotype may be displayed as is shown in Fig. 15, here an entity is related to the use case, it means that the entity

stores an authentication process or protocol, in other words, if the example is followed the <<IoTdevice>> named Device1 authenticates other entities e.g., sensors, actuators or tags. At this abstract level could be difficult to determine which hardware or infrastructure will address the authentication method, therefore, this representation fills the gap between the non-functional security requirements that in UML and SysML are not addressed and some automation tools or well-defined authentication protocols.

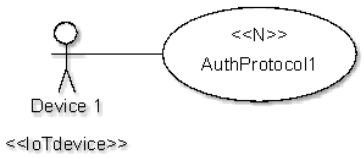

Fig. 15 <<N>> stereotype as a use case.

In static diagrams, the N element models a use case, class, software component, block (SysML) and object; on the other hand, in dynamic diagrams the N element depicts an activity or a state. In Fig. 16 the <<N>> stereotype and the corresponding extended metaclasses are shown, the UML notation is conserved, in the operations the principle functions described before are used as the minimum functionalities of the class, nonetheless the operations are open to any other functionality that could be related to the authentication module.

The N element has the *A* and *f* parameters as entry data, therefore the N element by itself or using another element of the nomenclature will determine *C* to obtain *l* [10]. *A* is the set of specific information with which entities prove their identities, *C* is the set of complementary information that is used by the system to validate authentication information*, F* is the set of complementation functions, *L* is the set of authentication functions to verify identity and *S* is the set of selection functions that enable an entity to generate.

With $l \in L$ and $f \in F$,

$$N(A, f) = l(A, C) = \{True, False\} \qquad \text{Equ. 1}$$

$$f(A) = C \qquad \text{Equ. 2}$$

$$l(A, C) = \{True, False\} \qquad \text{Equ. 3}$$

The <<N>> stereotype can be a class; hence it has instances (objects) which can be applied in a sequence diagram, object diagram and state diagram. The N element also can be modelled as a software component which can be allocated in different physical places, for this the UML notation is utilised. In Fig. 16 the metamodel for the <<N>> stereotype is displayed, the metaclasses extended are *class*, *UseCase*, *Component*, *Activity* and *State*. The <<N>> stereotype contains the three main operations described before and the assertion which is the result of the authentication process, therefore, it is recommendable to declare it as Boolean type.

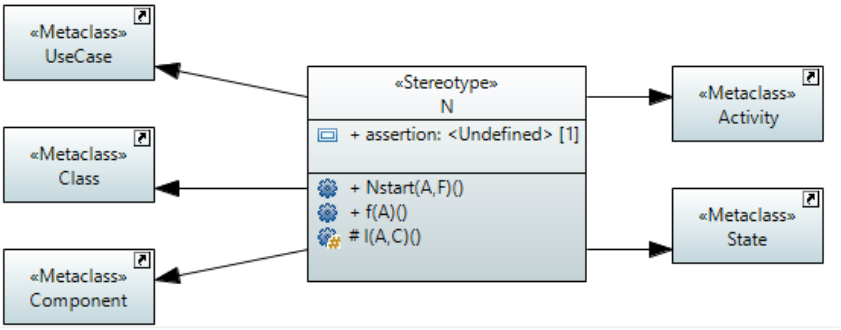

Fig. 16 IoTsecM <<N>> stereotype and metaclasses extended.

There are many approaches to guarantee actor's authentication, in [50] a survey of some authentication methods is introduced, nevertheless new authentication methods and new ways to exploit their vulnerabilities will be found. The function of the N element is to provide to the designers an abstract module that helps in the further implementation. This implementation can be modelled as well, using IoTsecM state machine diagrams where the life of an instance of <<N>> stereotype is described step by step.

4.2.2 Authorization: Z

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

Z authorizes or refuses a subject to access to a resource with some actions permitted related to its identity. From an abstract point of view its basic functional principle can be modelled like this [47]:

$$Z(s, r, o) \rightarrow \{true, false\} \qquad \text{Equ. 4}$$

$$with\ s \in S, r\ \in R\ and\ o \in O$$

S is the set of subjects performing the access.

R is the set of resources to be accessed.

O is the operation to be performed on the resource.

The functionality of the Z element is based on the AuthZ component in [13], so the recommendable data types for this element are:

Boolean: Z.authorize (Assertion, Resource, ActionType)

Boolean represents the result of the Z element decision, normally true or false (permit, deny). The Assertion depicts the accessing actor's information, typically are ID, certificate, Security Assertion Markup Language (SAML) assertion, Kerberos assertion, etc. The Resource indicates the asset to be accessed, for instance services or data. The ActionType depicts the action to be performed in the resource, for example read, write or execute.

In [51] the nomenclature of access control systems is defined, the authorise 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 Z element since these are the basic access control components which normally are present.

The Z element abstracts the main functionalities of an access control system in order to provide the system model with an abstraction of an access control.

Currently, there are two main access control mechanisms that could be suitable for IoT systems:

- Role-based access control (RBAC): In [52] the standard for the RBAC model is provided. The main elements of this model are Users, Roles, Objects, Operations and Permissions. A user is typically defined as a human being or a software agent, therefore it could be extended to the IoTsecM actors as well. A Role is a job function within the context of an organisation. Role refers to authority and responsibility conferred on the user assigned to this role. Permissions are approvals to perform one or more operation on one or more protected objects. An operation is an executable sequence of actions that can be initiated by the system entities. An object is a protected system resource. Two major relationships in this model are user assignment and permission assignment. User assignment relationship describes how users are assigned to their roles. Permission assignment relationship characterises the set of privileges assigned to a Role [53].
- Attribute-based access control (ABAC): the main feature of this mechanism is that it is not related to the identity of the actor directly, but on the attributes, he is presenting. This is for instance a permission based on the age of the actor. So, it delivers a high level of privacy for the user. However, it also requires a higher trust to the entity creating the assertion [47]. When the development team choose this kind of access control, it is recommendable to consider the modelling of the <<T&R>> stereotype which is introduced in section 4.2.7.

In an IoT system the access control policies are normally changing indirectly or in some cases they cannot even being changed therefore, access control policy will mostly be fixed since the deployment or even design time. The different actor's roles are fundamental to guarantee the firmware upgrade and the administration part of Z element (PAP) must provide the principle abstract operations.

IoTsecM provides an extension to depict whether an actor is authorised, it is writing a "Z" over the actor's head indicating that the current actor is or need to be authorised. In Fig. 17 there is an example where a sensor is an authorised actor and the Z is over its head. This is the first step in the design stage to map an authorization requirement within the IoTsecM notation.

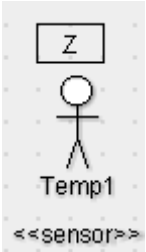

Fig. 17 Z element for authorised actors.

The functional principle described before is mostly related to Access Control Lists (ACLs) and its correct implementation, therefore Z element involves an authentication mechanism. The Z element must be able to map the certificates authenticated from N to some specific policies. A <<Z>> stereotype instance can be called from a <<IM>> stereotype instance if it is resolving a pseudonym.

In the IoTsecM use case diagram, an entity should provide the authorization service, which can apply the <<Z>> stereotyp to model it within a use case, e.g. in Fig. 18 an IoT device called central node has the use case Z wich depicts that this actor develop an authorization mechanism.

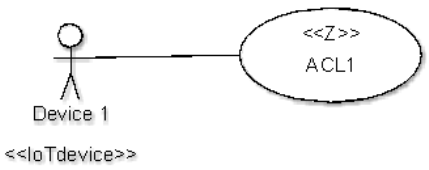

Fig. 18 <<Z>> stereotype applied in a use case

The Z element models the three main functionalities of an authorization infrastructure; therefore, the main operations of the PAP should be applied in the Z element. The IoTsecM profile, will allow to design these abstract operations within the <<Z>> stereotype, it is shown in Fig. 19 where the <<Z>> stereotype definition is shown and the metaclasses extended are *Class*, *Component*, *UseCase*, *Activity* and *State*. The main functionalities of a PAP are predefined in the operations: *Zstart(s,r,o), setPolicyRoot(), getPolicies(), setPolicy(), getPolicy(), addPolicy() and*

deletePolicy(). The <<Z>> stereotype includes three properties, two of them are Boolean which are *RBAC* and *ABAC* that define the control access type when true and the other property is an *ID*

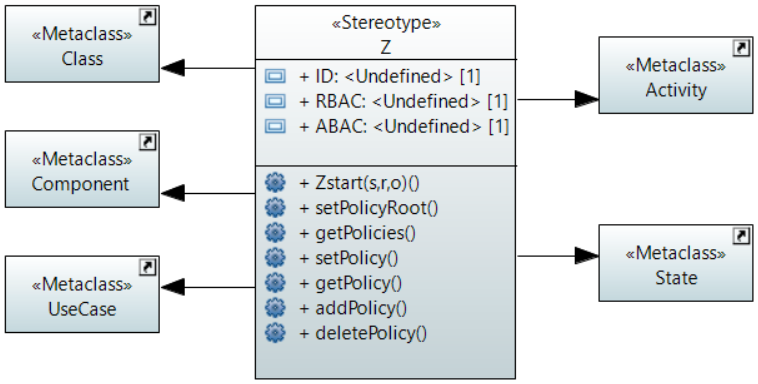

Fig. 19 <<Z>> stereotype definition.

4.2.3 C: Cipher and D: Decipher

Cryptography comes from two Greeks words, *kryptós* and *graphein* [47] which mean hidden and writing respectively, so it could be translated as "secret writing" and is the art and science of concealing meaning. It provides an indispensable tool set for securing data, transactions and personal privacy.

A cryptographic module should be able to offer the service of encryption, digital signature or message authentication code (MAC), in order to satisfy the confidentiality, authentication, integrity and non-repudiation features [47]. A good cryptosystem protects against:

- A *ciphertext only* attack, the adversary has only the ciphertext and its goal is to find the corresponding plaintext.
- A *known plaintext* attack, the adversary has the ciphertext and the plaintext, the goal is to find the key that was used.
- A *chosen plaintext* attack, the adversary may ask that specific plaintexts be enciphered. The attacker is given the corresponding ciphertext. The goal is to find the key that was used.

Cryptographic primitive types fall into the following categories [7]:

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

Cypher element means that a <<C>> stereotype requires the dynamic or static encryption of data, it should contain the encryption algorithms to achieve the encryption data following an algorithm given. This element can work with any other element which requires to encrypt data, therefore <<C>> stereotype is an abstraction of an encryption module, it provides symmetric encryption, asymmetric encryption, counter modes, hashes and digital signatures.

The C element receives the data, the key and the algorithm that will be applied, as a consequence it returns the encrypted data and the key used; for symmetric key it returns the only key and in the case of asymmetric key it returns the public key used for the encryption process. It is described in the next function:

$$Cout = Cin(Data, Key, AlgorithmID)$$

XOR operation is used in block chains and counter modes besides it helps to other system functionalities. Therefore, the C element includes the XOR operation.

$$XORout = XORin(Data1, Data2\ )$$

The counter modes make use of a counter, in these, the plaintext data is not actually encrypted with the cipher and key. Rather, each bit of plaintext is XOR'd with a stream of continuously produced cipher text comprising encrypted counter values that continuously increment.

$$CMout = Generate\_CypherText(Counter)$$

Although the encryption process is not always performed by just one module, it is important to add a counter mode which is able to produce cipher text.

In today's Internet threat environment, end-to-end encryption at the session and application layers is the most prominent due to severe data losses that can occur when decrypting within an intermediary. The security fixes often include building secure communication gateways (where newly added encryption is performed). In others, it is to tunnel the insecure protocols through end-to-end protected ones [7].

Symmetric algorithms consist of a ciphering operation using the plaintext or ciphertext input, combined with the shared cryptographic key.

The only asymmetric encryption algorithm in use today is RSA (Rivest, Shamir, Adelman), an integer factorisation cryptographic (IFC) algorithm that is practical for encrypting and decrypting small amounts of data (up to the modulus size in use).

In order to use block chaining modes, the <<C>> stereotype instance can be called as many times as the chain requires it, hence it can be easily added, since the C element is a meta-class which can be applied and added following the system requirements. In this way the cipher block chaining (CBC) are included. CBC is currently available as an option, for instance, in the ZigBee protocol (based on IEEE 802.15.4).

The <<C>> stereotype can be applied as a use case within the use case diagram to depict that a given actor has to encrypt data, or it could have an encryption module as well. In Fig. 21 the <<C>> stereotype is shown as a use case where an actor named CentralNode which is an IoTdevice has to encrypt data, this means that it requires the minimum encryption capabilities described before according to the system security requirements and computer power.

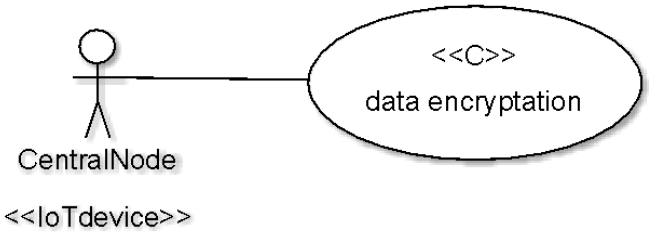

Fig. 20 <<C>> use case example.

The <<C>> stereotype extends the use case, class, component, device, activity and block (SysML) metaclasses. This stereotype includes the abstract operations previously described, such as *Cin(D, K, A)*, where *D* is the data to encrypt, *K* is the key and *A* is the algorithm required for the encryption process. The second operation provided by the <<C>> stereotype is a XOR, depicted as *XOR(D1,D2), D1* and *D2* are the data that will be XOR'd. The last operation is *GenerateCypherText(Counter)* which corresponds to the counter mode explained earlier.

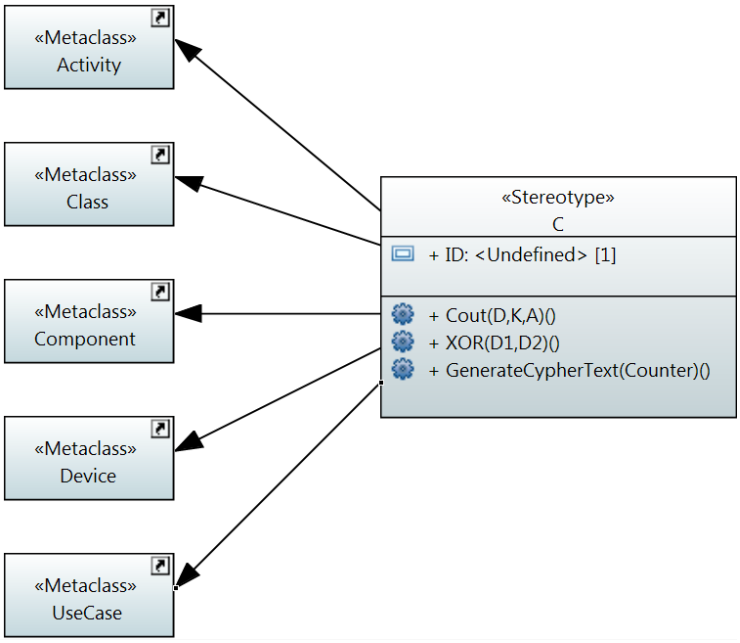

Fig. 21 <<C>> stereotype definition.

The main functionality of the D element is to decrypt the encrypted message; therefore, it must know the algorithm used in the encryption process, in that way it will be able to decrypt the information, it needs to know the key used to encrypt the data, therefore communication between the D element and the KM element 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 in order to customised as each IoT system requires it. The main functionality of the D element may be embedded in the C element, it is that to ease the model only the <<C>> stereotype can be applied assuming that it includes the next function which is actually the only D element function:

$$Dout = Din(Edata, Key, AlgorithmID)$$

Within the IoTsecM profile the D 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. 1, where an actor named "CentralNode" has a D use case. This means that such actor carry on the decipher process of a given data.

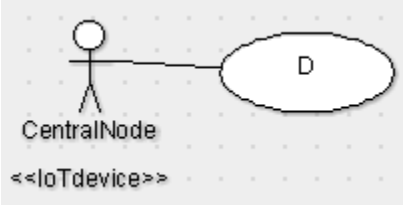

Fig. 1 D element as a use case.

The <<D>> stereotype is proposed to cover the decryption process in the analysis stage, therefore it extends the UseCase, class and component metaclasses from UML and on the other hand it extends the block metaclass from SysML. The only property defines is the ID and the operation defined is the Din operation which encapsulates the <<D>> function described earlier, in the Fig. 22 the <<D>> stereotype definition can be observed.

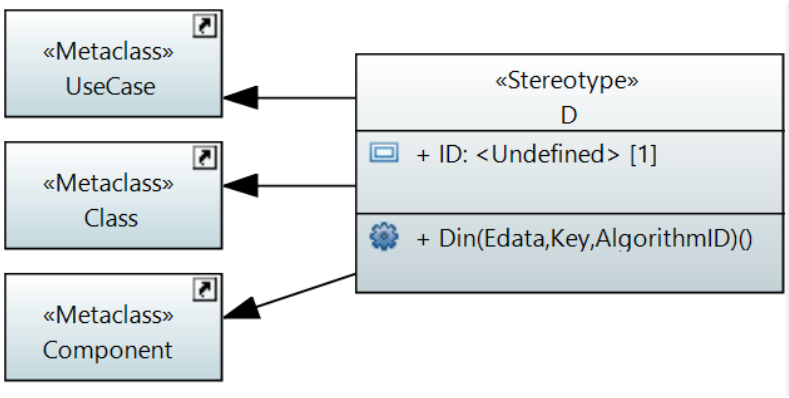

Fig. 22 <<D>> stereotype definition.

4.2.4 SS: Secure Storage

As it was 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. To protect sensitive data or meet privacy requirements, data access policy may be provided to enforce fine-grained data access rules for example requiring certain data or fields to be removed, obfuscated or redacted before distributing them to the data consumers for analysis or other uses [54]. The secure storage involves confidentiality and integrity of sensitive information stored.

Sensitive data can be protected applying encryption at the field, directory, record, file system or storage device level. Besides access control policies must be applied in order to guarantee that only authorised actor access the right information. In some cases the storage is performed at the cloud or in an extern device therefore, to protect data, provenance and privacy requirements must be attached to it.

The use of keys to encrypt and decrypt data blocks are used by cryptographic algorithms, nevertheless key managements imposes a hassle in IoT storage systems, in [55] a scheme without key management requirement is proposed applying Shamir's secret sharing scheme, since it is commonly used for small size secrets rather than large data management. Another light proposal is addressed in [56] where a combined secure storage and communication is proposed and validated,

there are others proposals for instance [57] where a confidential storage in sensor nodes within a WSN is introduced, it matches hardware capabilities with security requirements. The IoTsecM profile aims at covering the abstraction of a secure storage requirement, as it can be seem in those proposals in IoT systems it is not just to apply a cryptographic algorithm and keep safe the key, there are many other concerns such as power consume, latency, etc. therefore it is fundamental to regard the secure storage for IoT systems since the analysis stage.

IoTsecM profile proposes an extension to UML/SysML to depict the secure storage requirement, it is by applying the stereotype extension mechanism entitled <<SS>> which extends the constraint, link, association, communicationPath and property metaclasses from UML and requirement metaclass from SysML.

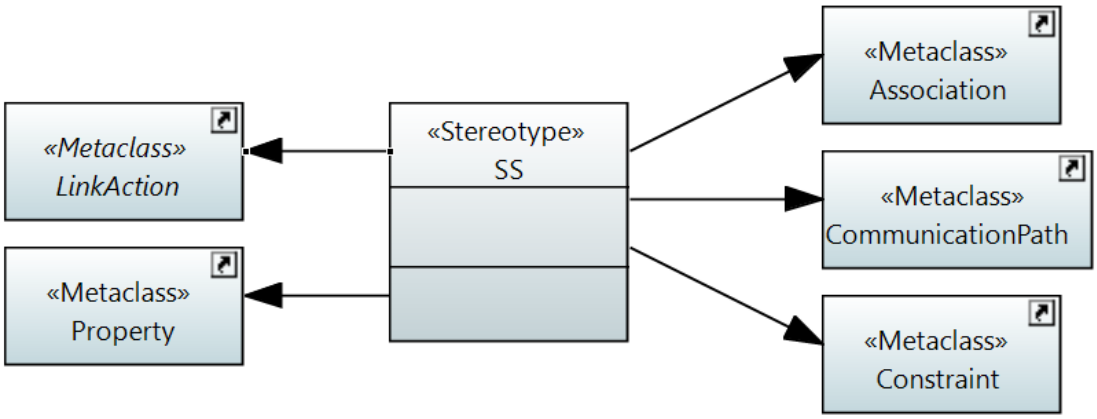

Fig. 23 <<SS>> stereotype definition.

4.2.5 SC: Secure Communication

In many cases at the benchmark of projects, the security mechanisms to implement or to consider are uncertain. As it is known in a software life-cycle the first approach that developers have with the system is how the client is figuring it out, they describe the system overview and then the first scenarios are obtained. At that point the developers are able to depict the first use cases and as a consequence the first security requirements in an abstract representation, as it was discussed earlies nowadays there are some approaches to achieve this, nevertheless within the use case diagrams it is not completely covered, there are some approaches as [16] where a UML extension

is proposed to elicit security requirements as misuse case, it is a threat or attacker model, nonetheless it does not cover the security design requirements that IoTsecM profile aims at covering. The secure communication ensures the data transmission between two authenticated, authorised and in some cases trusted entities, since sometimes even a partial leakage of information should be prevented.

When the secure communication requirement appear then some parameter should be addressed according to [13] 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 which would be running between the communicating nodes.The request should establish if 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.

This proposed extension is based on the UMLsec <<secure links>> stereotype [20] where the secure communication and information flow is regarded. In IoTsecM profile the <<SC>> stereotype is presented, it is a UML extension which depict that the communication between two entities needs to be secure, besides it is a constraint, this means that the security requirement must be attended in forwarded stages such as design, e.g. a certificate-based protocol may attend the {SC} constraint between two entities. Hence the <<SC>> stereotype extends the link, communication path and constraint metaclasses from UML and the requirement metaclass from SysML.

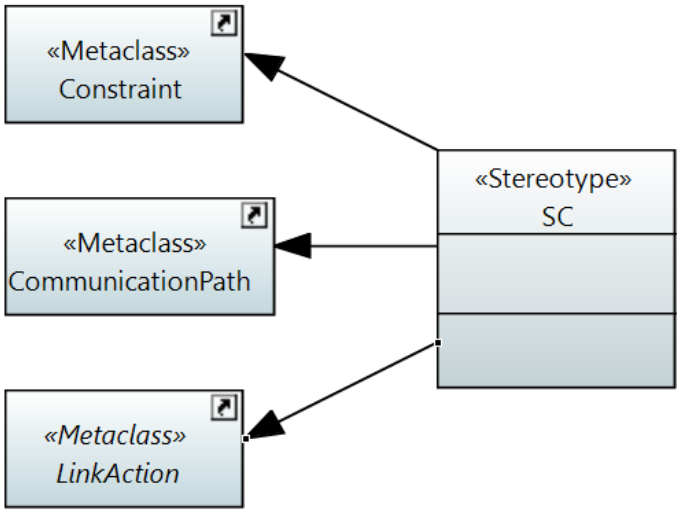

Fig. 24 <<SC>> stereotype definition.

4.2.6 KM: Key Management

A key management system enables and assist IoT assets in the rapprochement of secure communication or context, it is an integrated approach to generate, store and handle the keys within a cryptosystem. Efficient key distribution and management mechanisms are very important as well as lightweight ciphers. Besides the KM element is able to provide capabilities to assist the low-resource nodes in their operations, it depends on specific protocols and security mechanisms. The KM element should support the process of enhancing the security communication between a user and a service by setting up a tunnel [13] between gateways which is very useful for users and services running on low-resources 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 the known key management systems (KMS) do not apply in an IoT context, mainly because user and service are in different networks. According to [50] the KMS can be classified in four categories: key pool framework, mathematical framework negotiation framework and public key framework, nevertheless they argue that none of them is suitable for IoT environment, on the other hand the KMS protocols suitable for IoT systems are the Blom [58] which is a key predistribution method that allows any pair of nodes in a network to be able share a secret key and the polynomial schema, explained in [59], they are suitable for IoT systems since the computational power to run them is low in comparison to a Public Key Cryptography (PKC) operations. Nevertheless several countermeasures are required to manage device authentication and authorisation. In [60] a list of open source and property software for key management is provided.

The common behaviour of a KMS is to provide a trusted third party which provides the two entities the corresponding keys. The goals is to provide a secret key, e.g. if a user and an IoT device share different secret key and the objective is to provide a secret key to the user and IoT device share. It would follow the next simple protocol [61]:

- 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 decipher the message and uses $k_{session}$ to communicate with User. This is just the basis overviewed performance of a KMS, it has some vulnerabilities and, clearly, improvements explained in [61]. The KM element aims at describing and modelling the general behavior of a KMS, therefore some abstract operations need to be regarded such as:

- Key storage: A set of keys is securely stored for a subsequent use.
- Key generation: The KM element should be able to generate proper key when requested.
- Key exchange: In some cases, identical keys needs to be exchanged in others it requires to share only the public key.
- Crypto-shredding: Delete key data or revoke it.
- Key replacement: It should be able to replace a specific key.

The KM element was modelled applying a stereotype named <<KM>> which, as it should be known, is a UML extension mechanism. This stereotype models the operation above, it extends the next metaclasses: class, component, device and block (from SysML). It is depicted in Fig. 25, where the <<KM>> stereotype definition shown in a UML profile diagram.

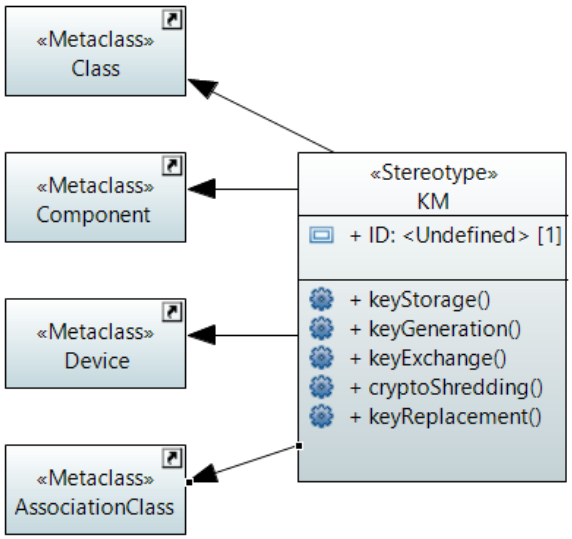

Fig. 25 <<KM>> stereotype definition.

As it is seen there is one attribute defined which is an ID and the abstract instructions which correspond to the KMS functionalities introduced earlier. The operations are: *keyStorage*, *keyGeneration*, *keyExchange*, *cryptoShredding* and *keyReplacement*.

4.2.7 T&R: Trust and Reputation

First the definition of trust and reputation are introduced. According to [62] trust is a particular level of the subjective probability with which an agent will perform a particular action, both before he can monitor such action (or independentlu of his capacity ever to be able to monitor it) and in a context in which it affects his own action. Reputation is an expectation about an agent's behaviour based on information about it or observations of its past behaviour according to [63].

The most comon functionalities of a trust and/or reputation models are described in [13] and introduced below:

- Gathering information: Collect behavioural information about the entities in the system, it may be obtained through several sources such as direct experiences with the targeted entity, neighbours, acquaintances, belonging group or organisation and even witness and pre-trusted entities.
- Scoring and ranking: Once the entity information has been gathered then it will be analised and scored. This computation could be achieved aplying fuzzy logic, Bayesian networks, analytic expressions or bio-inspired algorithms.
- Entity selection: The scoring data helps trusting entities to decide which entity interact with and which is not completely reliable. In the IoT landscape it would mean with which sensor interact.
- Transaction: Once the sensor to interact to was selected the transaction occurs between both entities giving a certain service.
- Reward and punishing: When the transaction is concreted, the client entity may asses that transaction in order to reward or punish the entity (in an IoT context, sensor), who provided the service.

The reputation needs to consider the low-computational power of some IoT entities such as sensor, hence such constraints need to be considered, hence being light, scalable, etc. There are some approaches to target the trust management in IoT systems such as [64] where a encounter-

based and activiy based trust management is proposed, in their proposal two sensors monitor and rate each other in order to exchange trust evaluation about other nodes, they use as reference parameters: honesty, cooperativeness and community-interest. Therefore a dynamic trust management protocol is capable of adaptively adjusting trust parameters. In [50] a brief introduction to many trust and reputation management systems is provided, those T&R management systems are classified according to the exploited technique such as: social networking, fuzzy technique, cooperative approach and identity-based method. They may be aplied in the design stage after the analysis of the necessity of a trust and reputation management existence is done besides of where to place it according to the respective IoT system architecture.

In IoTsecM profile the trust and reputation are regarded as an extension as well, where the extension mechanism applied is a stereotype called <<T&R>> stereotype the name should include the "or" word as well, nevertheles the name would not be very confortable to write, thus the <<T&R>> means trust and/or teputation. The <<T&R>>functionalities described above are depicted as operations within the stereotype definition they are shown in Fig. 26. The operations defined for this stereotype are gatherInformation, scoreEntity, selectEntity, transaction and rewardPunish, each one correspond to the general functionalities described above. The <<T&R>> stereotype includes two attributes the first is the ID and the second is the score of the trust and reputation processes.

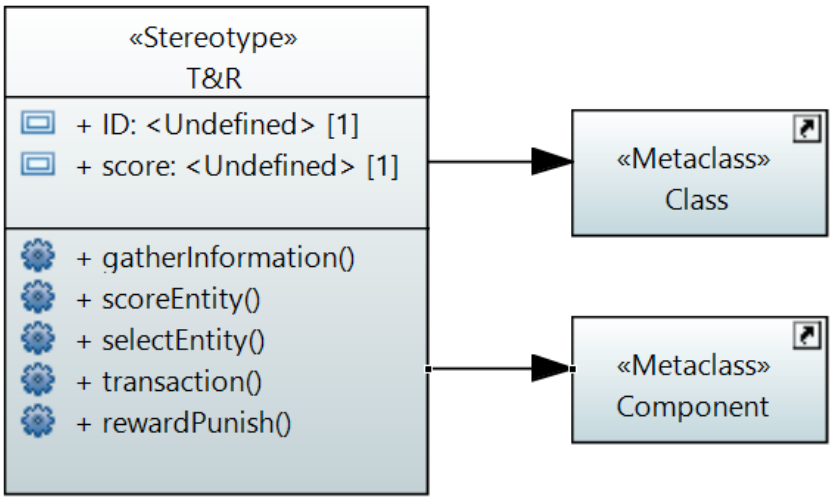

Fig. 26 <<T&R>> stereotype definition.

4.2.8 IM: Identity Management and Ps:Pseudonym

The IoTsecM identity management (IM) is based on [13] approach. In many IoT scenarios it is very important to protect the identity of users, actors, etc. therefore the information about the identity must be supplanted applying a pseudonym. This element is related to the <<PS>> stereotype, since it is the entity which handles the pseudonyms and system identities. The IM issues pseudonyms and accessory information to trusted subjects. This element protects the user privacy and service privacy.

A pseudonym is a temporary identity of imaginary subjects which include temporary credentials as well, access rights depends on the requesting subject therefore, a pseudonym can request another pseudonym.

The pseudonym generation may be depicted as:

$$createPs\ (s_1\ [s_1\ , s_2\ , s_3, \dots , s_n\ ], p\ \rightarrow\ s^*)$$

Where

s_i : is the subject or set of subjects.

s^*: is the requested pseudonym.

p: 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 less rights than the original subject, but it will never contain more rights. The expiration date possessed by the pseudonym is less or equal than the subject's one. The request of a pseudonym should be only requested by a secure channel.

The IM element is the only one that keeps the relation between the pseudonyms and the subjects. Another functionality of the IM is that when creating a new pseudonym it must update the access policies in Z element and as a consequence associate a new address to the ID.

Nowadays X.509 certificates only provide a benchmark for building an IoT authentication and authorisation capability. Vendors support Identity Relationship Management (IRM), organisations as GlobalSign have begun to build these concepts and support delivery of high volumes of certificates [7].

Two stereotypes are defined to depict the pseudonyms concern, the <<Ps>> stereotype is an abbreviation of the pseudonym word and it depicts the requirement of an actor of a pseudonyms, for instance if an <<IoTdevice>> entitled as "node1" requires a pseudonym, then the <<Ps>> is depicted as shown in Fig. 27, this will help to identify whose actors protect their identity and as a consequence the actor's privacy.

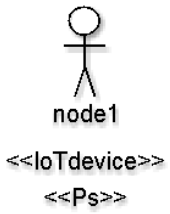

Fig. 27 <<Ps>> stereotype example.

The <<Ps>> stereotype extends the actor metaclass, the constraint metaclass and the requirement metaclass from SysML. In Fig. 28 the <<Ps>> stereotype definition is shown.

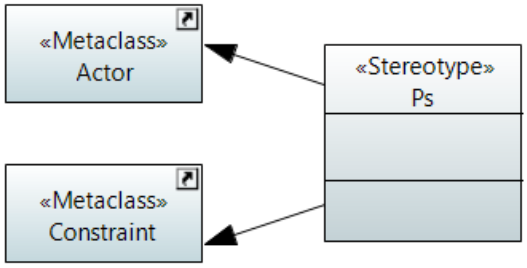

Fig. 28 <<Ps>> stereotype definition.

For the <<IM>> stereotype is applied to depict the identity management mentioned earlier, it extends the class, component, activity and association metaclasses. It defines three main operations: verify which verify the assertion in the corresponding <<N>> instance, the changeRightsInZ operation which change the root identity permissions and assign the previous ones to the pseudonym generated regarding the constraints mentioned before, the last operation defined is creates which receives the current identity and some specifications, as it can be noticed it models the previous behaviour function. The <<IM>> stereotype has two attributes, the first one is the ID attribute and the second one is a table which contains the links between the original root entities and pseudonyms generated, it cannot be queried by any external entity, module, component, etc.

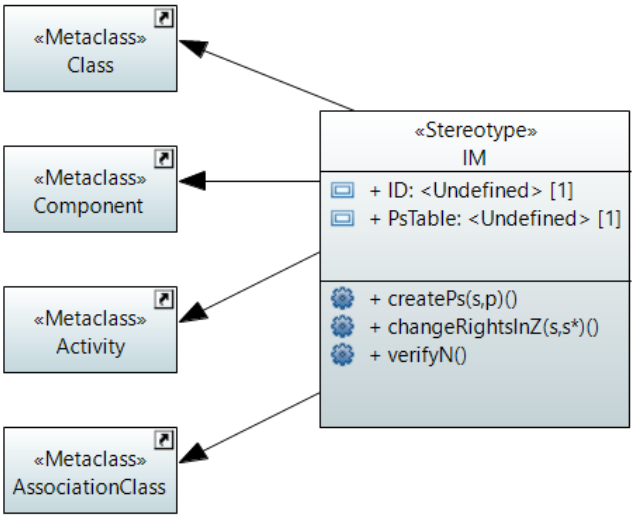

Fig. 29 <<IM>> stereotype definition.

4.2.9 CA: Certification Authority and RA: Registration Authority

It is a trusted entity which is responsible of issue and revoke digital certificates, using a digital signature, where a public key cryptographic algorithm is applied. The certificates include numeric IDs and needed passwords besides it makes available the verification process to validate the provided certificate. The CA legitimates to third entities who trust in the CA certificates the relation between the actor identity and its public key. Although there is not a normalised process to trust in

CA, it is a fundamental concept for the correct performance, then the entities who request a certificate from a CA must trust on it.

An AC is normally applied in a public key infrastructure (PKI), where prior to 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 organisation 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 digital certificate according to the standard ITU X.509 is illustrated in Fig. 30.

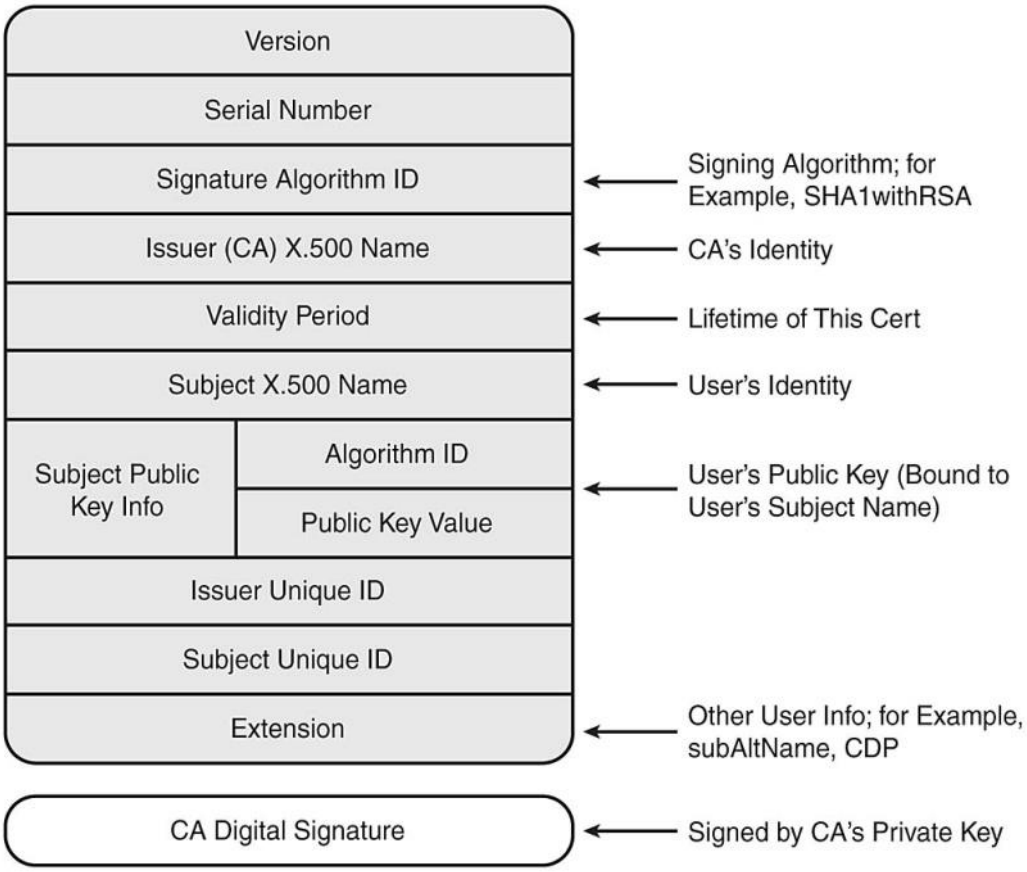

Fig. 30 Certificate structure according to the ITU standard X.509.

The CA services are commonly used in digital communications in TLS protocol, e.g. to securitize the web communications (HTTPS) or email communications (SMTP, POP3, IMAP)

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

In the IoTsecM profile the CA is regarded as an extension to UML/SysML, this modelling artifact is extended by a stereotype which is named <<CA>>. The <<CA>> stereotype is thought to be applied as class, component and device in UML, on the other hand in SysML it extends the block metaclass. The functionalities of the <<CA>> stereotype are: issue a certificate, verify the entity (this operation is with the registration authority) and revoke the certificate, in order to model this abstract functionalities three operations were proposed, which are *issueCert*, *verifyEntity and revokeCert*, each one corresponds to the previous functionalities.

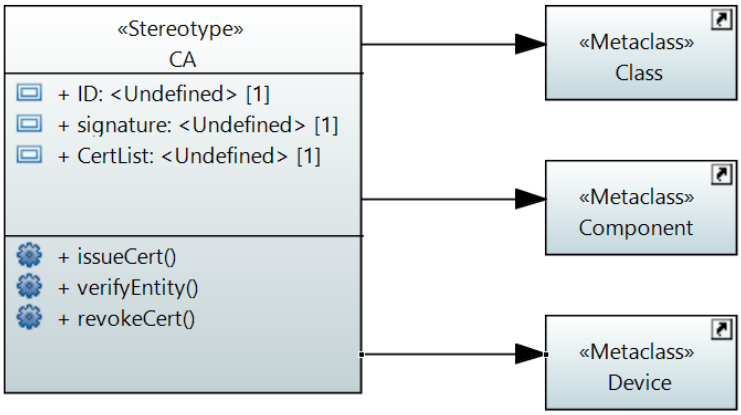

Fig. 31 <<CA>> stereotype definition.

<<CA>> stereotype is able participate to verify 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 Z, T&R and N rely on this element to link their activities to the correct subjects.

The registration authority controls the certificate generation, it realizes the certification petition and save the corresponding data, it is normally encapsulated in the CA, nevertheless as IoTsecM aims at covering as many architectures and configurations as possible it is regarded as another element which is 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 the class and component UML metaclasses and block from SysML metaclass. It does not extend the device metaclass since it is normally encapsulated in CA element, nevertheless in order to obtain a greater felixibility in analysis and design stages, it is regarded to extend the class and component metaclasses. The operations tha <<RA>> includes are registerActor, verifyActor and sen2CA, which models the functionalities introduced above. The attributes that <<RA>> stereotype includes are and ID and a register which is the record of the certificates to be issued.

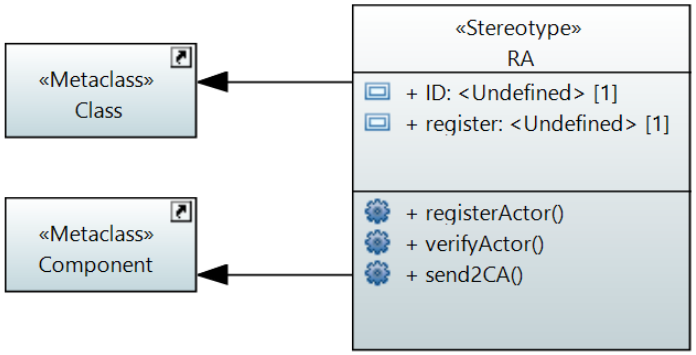

Fig. 32 <<RA>> stereotype definition.

4.2.10 TP: Tamper Protection

The IoT environment involves many scenarios where physical entity to be observed is located at a not reachable and exposed place, thus it would be expensive and hard to protect it with infrastructure or vigilance. The information safeguarding from disclosure implies understand the physical security needs of an IoT system. According to [49] the physical security affects architectural design, polices and even technology acquisition. The solution to the physical threats is to attempt to drive IoT device procurements that include physical tamper protection.

Due to IoT devices and sensors may be deployed in remote places an attacker might tamper with them and capture them in order to see, delete or modify information. The attacker could extract cryptographic secrets, modify programs or replace them with malicious nodes. Therefore, a tamper resistant packaging would assist the defence against the existing threats [65]. The tamper resistant package would mitigate the physical attacks which threaten the confidentiality, integrity and vulnerability information as well as actor's privacy.

In many scenarios the sensors will not include a tamper-proof hardware, therefore an attacker might gain access to them and impersonate them or see the information stored on the sensor nodes, some techniques such as shrinking the sensor buffer or a wrap protection should be regarded, nevertheless if an attacker gain access to the sensor node it would be catastrophic for instance in the case that a RSA private key were contained there without tamper protection (implemented in software or deployed on hardware). In [66] a tamper-proof embedded chip (TPM) is proposed to provides tamper proof generation and storage of RSA keys as well as hardware support for the RSA algorithm, besides the TPM certificate and the trusted CA certificate must be stored on the publisher prior to deployment.

Another factor as important as the tamper-proof mechanisms is the tamper-detect, in [67] is mentioned that systems should provide tamper-evident environments such that any physical tampering or software tampering by an adversary is guaranteed to be detected.

In IoTsecM profile the tamper protection is regarded. A UML/SysML extension is proposed named <<TP>> stereotype which means tamper protection and represent the security requirement of tampering resistance, which means that an entity containing the <<TP>> stereotype must be

consider since analysis and design stage a tamper protection in order to do not be logically or physically altered.

The <<TP>> stereotype provides a security requirement of tamper protection to entities and system components (hardware and software), thus it extends the constraint and property metaclasses from UML, as well as requirement metaclass in SysML. Therefore the <<TP>> stereotype is able to depict constraints in UML diagrams in order to indicate that a given entitiy requires to be tamper-proof, besides it is a property hence it can be displayed within a class property, in order to indicate that such class or class property requires to be protected to tampering. The <<TP>> stereotype is part of the IoTsecM profile and as a consequence it may be used in a secure development design process.

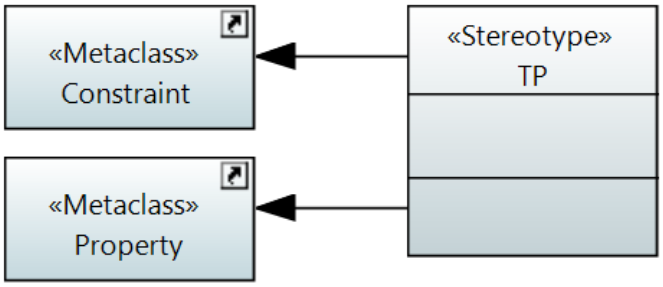

Fig. 33 <<TP>> stereotype definition.

4.2.11 BM: Behaviour monitor

The IoT security requirements involve mainly the first defence line, which is normally established by the <<Z>> and <<N>> stereotype instances, these security mechanisms provide security to some part of the system, nevertheless there is not a system without vulnerabilities, due to inside or outside intruders that may exploit wireless communication protocols such as an attack from inside the 6LoWPAN network and from the Internet. Therefore, another defence line is needed where a security control be able to monitor the system behaviour in order to detect the malicious behaviour of the then report it and in some cases act on it. In a passive systems, the behaviour monitor detects a possible intrusion, store the information and sends an alert signal that is stored in a database. In

a reactive system the behaviour monitor reacts to the suspicious activity reprograming the firewall, if it is the case, or updating the policies within the <<Z>> stereotype in order that it can block the traffic which comes from the attacker. The analogy to the classical security mechanisms are: Intrusion Detection system (IDS) and Intrusion Protection System (IPS). The IDS monitors the operations in a network, alerting the system administrator when it detects a security violation. According to [68], currently, there are no IDSs that meet the requirements of the IPv6-connected IoT since the available approaches are either customised for Wireless Sensor Networks (WSN) or for the conventional Internet, besides applying traditional IDS techniques to IoT is difficult due to its particular characteristics such as constrained-resource devices, specific protocol stacks and standards.

The IoT systems are composed of devices with resource constraints in many cases, hence one concern to implement a IDS in an IoT system is to find nodes which comply with the computational resources to support a IDS, besides it must be located in a place where the IDS be relevant, thus it is not an easy work to find where to implement an IDS when the systems is being deployed, therefore from the first design stage they should be considered, in order to provide an architecture and devices which support the behaviour monitor.

Within IoT systems, the behaviour monitor may be located in the border router, in one or more dedicated hosts, or in every physical object [69]. The advantage of placing the IDS in the border router is the detection of intrusion attacks from the Internet against the objects in the physical domain. In [69] taxonomy of IDSs for IoT systems is provided besides of a survey of the principal approaches to IDSs within the IoT or technologies related to them.

The IoTsecM profile proposes an abstract module which allows the behaviour monitoring of some or specific system part. The analysis of the network traffic, the port scanning and the malformed packets are some of the functionalities of this element. This approach addresses the behaviour monitoring security requirement proposing a UML extension mechanism, which is a stereotype named <<BM>> which encapsulates all the semantic described in this section, besides it can be properly placed since the analysis stage, then, if the stakeholders and developers decide it so, the devices and components will be bought in order to guarantee that the chosen IDSs can run without resources concerns. In order words once the decision of an IDS in the system is taken, then the hardware and software to support can be proposed.

The <<BM>> stereotype addressed the necessity of a behaviour analyser in the system extending the metaclasses: component and device component from UML and the block metaclass from SysML. The <<BM>> stereotype defines two properties: an ID and a BMType whixh can be an IDS or IPS. There are three operations defined which correspond to the behaviour of the <<BM>> stereotype, these operations are: *analyzeTraffic*, *scanPorts* and *reviwePackets*.

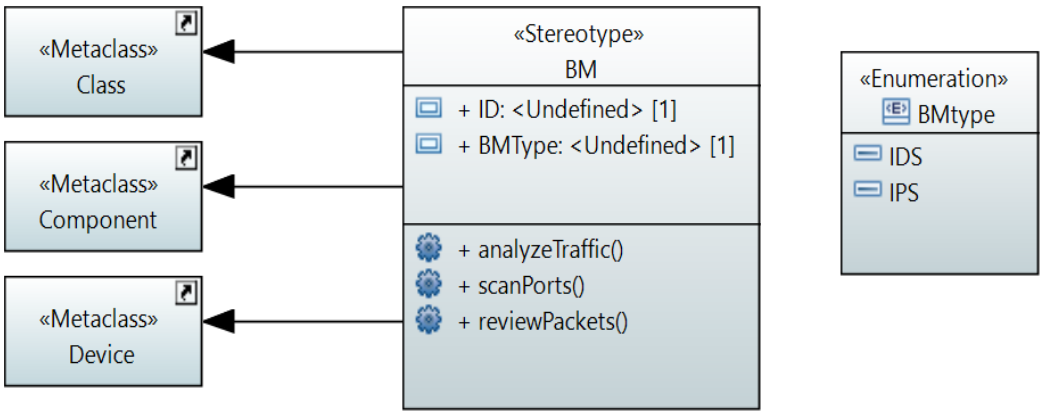

Fig. 34 <<BM>> stereotype definition.

5 APPLICATION OF IOTSECM PROFILE AND RESULTS DISCUSSION

The IoTsecM profile was presented in chapter 4, in this chapter the profile application is described in order to validate its usability in a real-life IoT system. The IoTsecM profile was applied in two systems, the first one is related to autonomous vehicles and the second one is related to an m-health system.

As it was described IoTsecM profile addresses the designing and modelling of IoT systems considering a security architecture, it helps to depict the system security concerns and nevertheless, the security mechanisms should be in an accurate place, in order to protect the system against a real threat or attack. Once the possible attacks over the system are identified then the developer will be able to figure out a way to provide protection or countermeasures against those attacks, therefore, they would be able to find the right place for the right countermeasure for a particular attack or threat.

Threat modelling can be achieved by different ways, there is not a unique methodology which helps to mitigate the system risks. The main threat modelling objective is to know the system threats and vulnerabilities, which surely would be exploited by a motivated attacker if countermeasures are not there to prevent them. More about threat modelling can be found in [70]. Microsoft proposes another approach which uses multiple steps to determine the severity of threats, this approach is referred to as Microsoft SDL [71]. The approach applied for the threat modelling is described in each study case.

In this chapter two study cases are presented, threat modelling and attack trees are obtained for each case. Based on the analysis performed, and using the IoTsecM profile proposed, security countermeasures are derived and represented in the IoT system's architecture.

5.1 AUTONOMOUS VEHICLES

The first case of study considered to verify the applicability of IoTsecM is within the smart cities domain, and specifically it is related to an autonomous vehicles system. In the system, not only autonomous vehicles are considered but also the interaction of these with other assets such as city infrastructure sensors and traffic lights. This case of study is a real-life project named Flourish [71]. It runs from June 2016 to May 2019 and the main project's objective is to find innovative solutions related to customer interaction, connectivity, data analytics and safe design for collaborative autonomous vehicles (CAVs). The system overview is shown in Fig. 35.

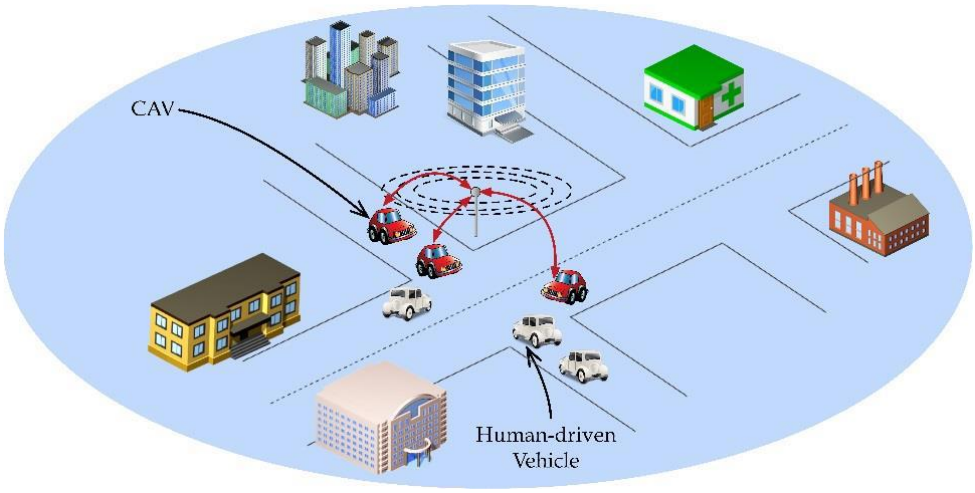

Fig. 35 Flourish project overview.

The Flourish project covers many knowledge areas; however, the main topic concerns the design of secure CAVs. The objective of the introducing IoTsecM into Flourish is to provide an application architecture where the security mechanisms and controls are depicted and modelled in order to enable secure, trustworthy and private technology within the CAVs and the whole infrastructure.

The Flourish project undertakes simulated and real world tests of communication systems in order to develop products and services for the market. Hence, IoTsecM focuses on the security modelling design process for communications and privacy issues. The IoTsecM extensions provide a

notation and semantics which model and depict the security concerns in the system architecture model.

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

In order to identify the security concerns a first approach was developed by the Flourish team, where general security requirements were regarded in order to propose the first application architecture. However, they just considered general attacks such as Denial of Service (DoS), jamming, Sniffing, tampering, spoofing, etc., but these attacks were not related to the system architecture. In addition, a threat modelling or any method to model the attacks or attacker was not developed. This kind of analysis would allow identify vulnerabilities and new attacks may be identified, therefore a MBSE process is needed for this work, besides it would validate the security mechanisms they proposed and would add others. Furthermore, the use of IoTsecM profile will make possible to identify new security requirements, therefore the IoTsecM helps to place rightly the security mechanisms within the system architecture and identify and depict the previously proposed security mechanisms working together with the new ones.

Threat modelling helps to identify threats and threat sources, it provides a methodological approach to perform a security evaluation of a system.

In this work, the process followed to perform the threat modelling and security countermeasures analysis and design is based on [49]; however, this process was customised and extended in order to add the countermeasures modelling, the process followed is summarised into the next steps:

- Identify the assets
- Create an IoT system architecture overview
- Decompose the IoT system
- Identify threats
- Document threats
- Propose counter-measures for each threat
- Propose a system architecture depicting security countermeasures

As it was written earlier, the Flourish project involves autonomous vehicles communicating to each other and with human driven vehicles (HDV) and to road side units (RSU). This communication is referred to as V2X, vehicle to everything. The system architecture overview consists mainly in CAVs, which are autonomous vehicles that are 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 in order to communicate to the CAVs and to diverse processing centres. Therefore, the system architecture overview consists of those three assets categories:

- CAV
- RSU
- Processing nodes

Although the system architecture looks simple, it is not, it consists of many different assets, and those assets were obtained following the scenarios described by the Flourish team. Those scenarios are not described in this work since they are part of the internal deliverables and must remain confidential; nevertheless, the assets list is shown below, since the assets identification allows an understanding of what must be protected. The assets are system components which are of interest to an attacker; therefore, they can be hardware, software, physical entities or even humans. The assets were obtained analysing the scenarios provided by the Flourish team, who described each scenario as general system use cases. The assets list is shown in Table 5 Flourish assets.

Table 5 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 road side unit
BBR data	Data created by the LIDAR about what it observes
BBR+ feed data	The same BBR data plus new observations
RSU instructions	Data send from RSU to CAVs
Traffic signal control data	Data sent from a control centre to control the traffic signals

Optimisation of traffic signal control data	Optimisation of data sent from control centre to control the traffic signals
Control node	Is the node dedicated to elaborate CAVs manoeuvring actions when receives information related to HDV intentions and CAVs driven intentions.
CAVs manoeuvring actions data	Is the main result of the control node processing unit, it indicates to CAVs the new manoeuvring actions
Motion description of HDV data	It is the data within the RSU which describes the motion description of HDV
Carer	Is the person dedicated to activate the point in order to establish a special zone which is a restriction zone for special requirements such as slow traffic
Point to activate especial zone	Is the point dedicated to send the request for a special zone
Request for exclusion zone data	Is the request send by the carer through the special point to the RSU in order to broadcast it to the CAVs
Special zone information	Information about the special zone requested such as duration, space restrictions, etc.
Network Rules Engine (NRE)	It is the processing node dedicated to provide routing advisory and communicates to the Network AI Unit
Congestion reduction data	It is 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 and it is a prediction of the congestion on roads
Routing advisory data	It is the data to advise the CAVs with new routes in order to reduce the traffic
Control room	It is another processing node which gives priority to roads
Control algorithms	They are the algorithms that are contained within the control room and are used by it as well
Instructions data	It is the result data delivered by the control room and sent to CAVs through the RSU
On board sensors	They are sensors located within the CAV 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

Once the assets were identified the next step, following the threat modelling described earlier, is to create a system/architecture overview, this step affords the IoT system functionality. For this part the scenarios provided by the Flourish team were very useful as well, since they described the system use cases and therefore the system functionality; however, the architecture view of the system regarding the security issues is proposed in this section where the interactions and system assets appear together conforming the Flourish IoT system.

The system architecture is depicted in UML class diagram, it considers the assets identified before and their connections. As it can be seen in Fig. 36 there are three main components within the architecture: CAVs, RSU and intelligent transport systems (ITS) central station.

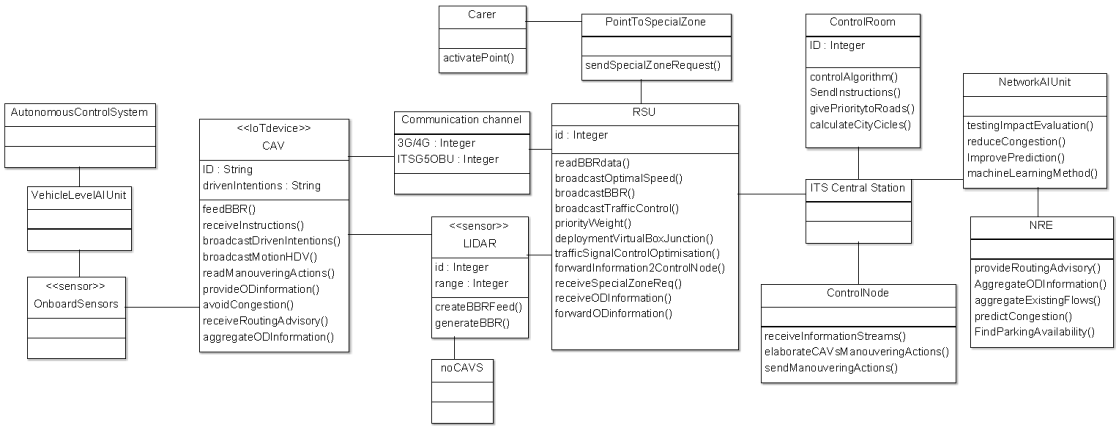

Fig. 36 Flourish architecture overview.

The CAVs holds the on-board sensors, the vehicle level AI unit and the autonomous control, besides it contains some attributes such as an ID and its driven intensions. The operations that the CAVs holds are: *feedBBR, receiveInstructions, broadcastDrivenIntensions, broadcastMotionHDV, readManoeuvringActions, provideODinformation, avoidCongestion, receiveRoutingAdvisory* and *aggregateODInformation*, each one of these instructions corresponds to one functionality described in the scenarios, e.g. the broadcastDrivenIntensions operation correspond to the use case of manoeuvring collaboration where CAVs must broadcast its driven intentions to other CAVs in order for them to correctly react to the new movements and even predict new driven intensions. Due to confidentiality issues the scenarios are note completely described in this work.

The communication channel between CAVs and RSU may be achieved through two ways, the first one is by 3G/4G technology and the second one is by the ITSG5OBU, which is related to the infrastructure proposed by the Flourish team, this two ways allow the CAVs to send and receive data from the RSU.

The RSU opperations are: *readBBRdata, broadcastOptimalSpeed, broadcastBBR, broadcastTrafficControl, priorityWeight, deploymentVirtualBoxJunction, trafficSignalControlOptimisation, forwardInformation2ControlNode, receiveSpecialZoneReq, receiveODinformation* and *forwardODinformation*. The principal RSU functionality is to receive information from the 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 asset, the operations defined to model the control node behaviour are: *receiveInformationStreams, elaborateCAVsManoeuveringActions* and *sendManoeuveringActions*. Those operations are related to the collaborative manoeuvring actions, the actions are calculated in the control node and sent to the RSU in order for the CAVs to receive them and act according to them.

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

The network AI unit is modelled with the *NetworkAIunit* class therefore, its operations correspond to the network AI unit behaviour. The control room is modelled with the *ControlRoom* and its operations (*controlAlgorithm*, *SendInstructions*, *givePriorityRoads* and *calculateCityCicles*) are focused on give priority to certain roads and send instructions to the RSU. The special zone point is modelled 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 behaviour. The carer is the person who activate the special zone through the point dedicated to activating it, this asset is modelled by the *Carer* class and it includes one operation *activatePoint*. LIDAR is the asset which monitors the CAVs and HDV, it obtains and creates data about the movements, the operations defined for the LIDAR class are *createBBRFeed* and *generateBBR*. NoCAVs is the class which models the HDV vehicles and other motioned objects that the LIDAR may observe.

The Flourish architecture includes all the classes described before, once the system architecture is already understood then it is time to identify the threats. In order to address the threat identification an intuitive and graphic notation is needed, since at this stage the system has not been deployed yet, and all the analysis is done with graphic representations. Therefore, attack trees diagrams are proposed to address the threat identification.

AAttack trees are an orderly and sequential way of describing the sub-attacks to violate a system, they are a useful tool to conceptualize and visualize the possible attacks, where the designer must put himself in the attacker's shoes to devise all the different ways in how an asset can be violated. This analysis results in the underlying root causes of attacks, allowing the analyst to create

attacker profiles, in order to 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, as far as possible, the entire attack surface and it may be difficult to figure out all of them. It is recommendable to do it within a working group where at least two people can build it together. A tool is needed for this purpose, in this work the one named SecureItree, an attack tree modelling tool built by the Canadian company *Amenaza* (the Spanish word for threat) [72], was used . In this tool the root node represents the end objective and the children nodes depict the different sub attacks in order to accomplish the overarching goal. The nodes can be AND operator, OR operator, or a LEAF. The AND operator means that all of the children nodes are needed to accomplish the parent node, on the other hand, the OR operator means that any of the children nodes satisfy the parent node.

One of the underlying concerns for the Flourish system is the communication flow; hence, the first threat to be modelled is related to the communication between the CAVs and the RSU as depicted in Fig. 37. 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 regarded to achieve the goal are:

- Jamming data from CAVs to RSU: This kind of attack is very common to compromise a wireless environment such as the communication between the RSU and CAVs, the goal is to drop the signal to a level where the communication is interrupted. Typically the older wireless area networks are the most vulnerable to the success of this kind of attack, since the actual networks are able to adapt to unintentional or intentional interference. The countermeasure proposed for this attack is an intrusion prevention system (IPS), since it should be able to detect the presence of any unauthorised client device.
- Turn off power: this attack is related to turn off power from the car, there are 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 doors locks. The other way to turn off the power is if a malicious passenger gets in the car, this kind of attack may be achieved with social engineering, where the malicious passenger obtains authorised 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 send many requests to the CAVs or to the RSU in order to make them attend just the false and mal formed requests whereas they deny any other request even the authenticated requests. The countermeasures against this attack is an intrusion detection system (IDS) or IPS.

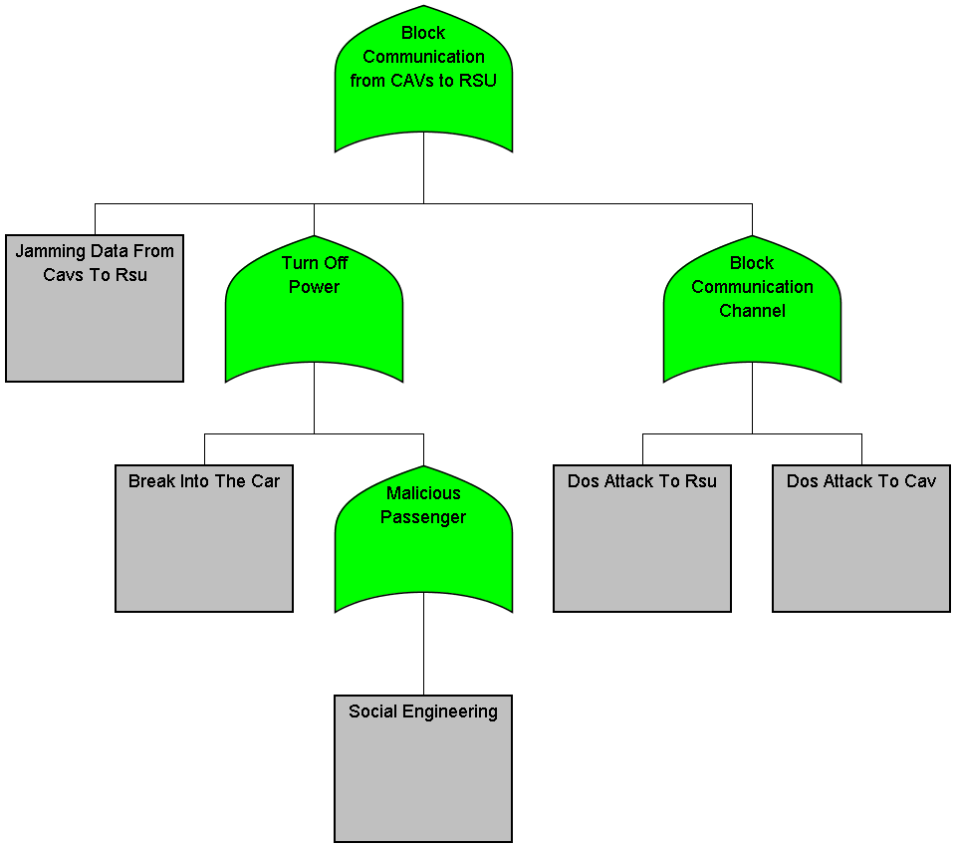

Fig. 37 Block communication from CAVs to RSU attak tree.

The next threat is the spoofing of BBR data, this means that an attacker is able, somehow, to masquerading himself as another actor in order to falsify data, in this case the BBR data which is 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 this attacks would be able to thwart the complete vector attack. . The attack tree considered for this threat involves the next sub attacks:

- Tamper the on-board sensors: This attack means that the on-board sensors are manipulated by an attacker who is able to change the data which is reading the sensor in order to create false information and as a consequence 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 which at this point 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 a man in the middle (MITM) attack. If an attacker is able to impersonate the CAV sensor node, then it can receive and change the on-board sensors data. Therefore, the countermeasure for this attack is the authentication of the on-board sensors, in order to guarantee that sensors are who they claim to be.
- Create a fake RSU: This attack consists in create a false RSU in order to perform a MITM attack, in this way the LIDAR would not be able to identify the false RSU, would trust on 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 a lot to mitigate this kind of attacks.
- Create a fake processing node: It is very similar to the last attack. However, the MITM attack here 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 the LIDAR, this means that a well-motivated attacker performs a physical tamper to the LIDAR, in order to change the data which is about to send. Because of this potential attack a tamper protection for the software and hardware is required in the LIDAR.

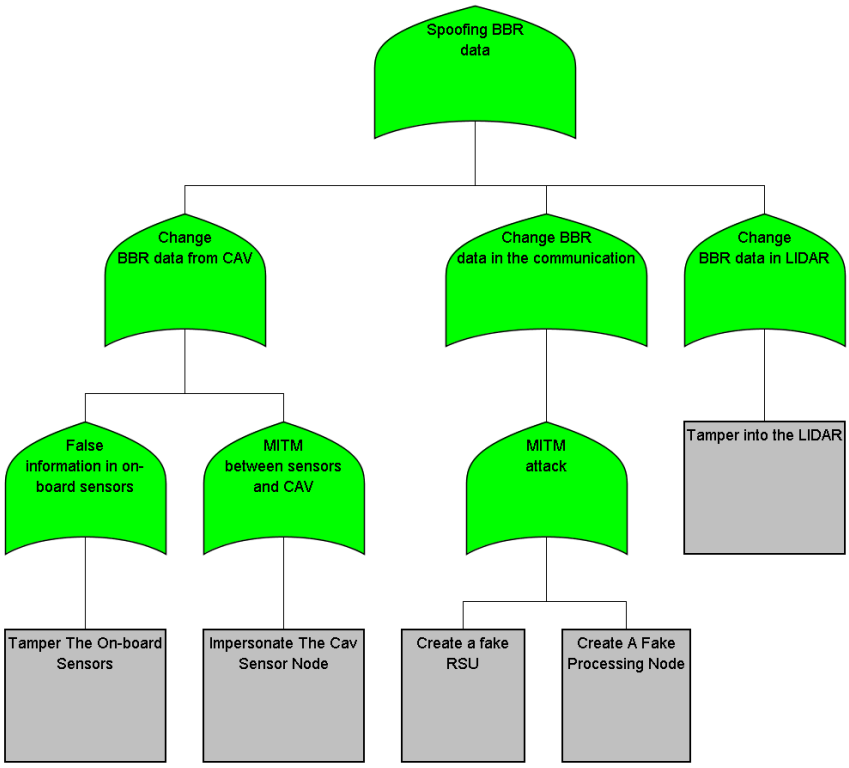

Fig. 38 Spoofing BBR data attack tree.

The next attack tree documented regards the carer impersonation threat and it is depicted in Fig. 39. This threat involves the carer which is the person that activate the point in order to request a special zone. Therefore, the carer impersonation implies that the attacker is able to activate at least one especial zone with the goal of alter the traffic flow. As in the last attack tree the presentation of the leaf attacks is described aiming at proposing countermeasures, since the leaf attacks are mitigated then the corresponding vector(s) will be thwart.

- Social engineering: An attacker provides authentic credentials to the activation point, with the application of social engineering. The countermeasure for this attack is the activation point access policies strengthen and the carer authentication.
- MITM attack between the activation point and the RSU: An attacker is placed between the activation point and the RSU, he captures the activation point signal and he is able to send it at any time. Therefore, he creates a fake RSU. The countermeasure against this attack is to authenticate the activation point.

- Fake activation point: Place in the right location a fake activation point in order to avoid the special zone request. The authentication of the activation point is the countermeasure to prevent this attack.
- Study the carer's routines: This attack is part of a plan to kidnap the carer, this attack resides out of the scope of the Flourish project, nevertheless the recommendation for the carer is that all its personal information should be encrypted in order to do not allow the attacker to read his routines, habits, etc.
- Brute force: to force the carer to activate the special zone, this attack also resides out of the scope of the security concerns for the Flourish system.

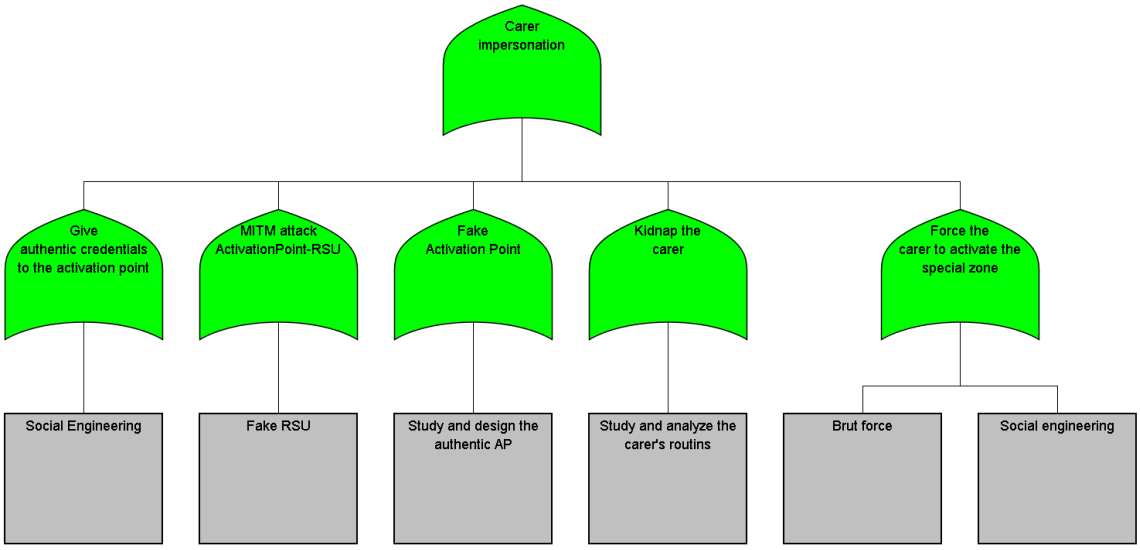

Fig. 39 Carer impersonation attack tree.

The next attack tree regards the jamming of the RSU communication, in this case the attacker objective is to block all the RSU communication and this means that it would not be able to transmit or receive any data; this attack tree is displayed in Fig. 40. The risk of the success of this attack is that, one part of the system would not work properly since the RSU is the intermediary between CAVs and almost any other asset within the system. The sub-attacks considered to reach this threat are:

- Jamming radar implementation: This radar creates a signal of the same RSU signal frequency in order to interrupt the RSU communication. The countermeasure is an IDS or IPS running in the RSU aiming at identifying the jamming radar and block its interruption.
- Create false CAVs requests: This attack is a DoS attack, it is targeted to the RSU, consisting in creating false CAV requests. The countermeasure against this attack is an IDS or IPS in the RSU.
- Shut down the RSU: The way to shut down the RSU is gaining access to the place where it is located. Therefore, the countermeasure against this attack is a tamper protection for the RSU.
- Flashing the RSU data: The way to reach this attack is split in two ways, the first is breaking into the RSU and the second is with remote control. For this attack the countermeasure regarded is a tamper protection for the RSU.

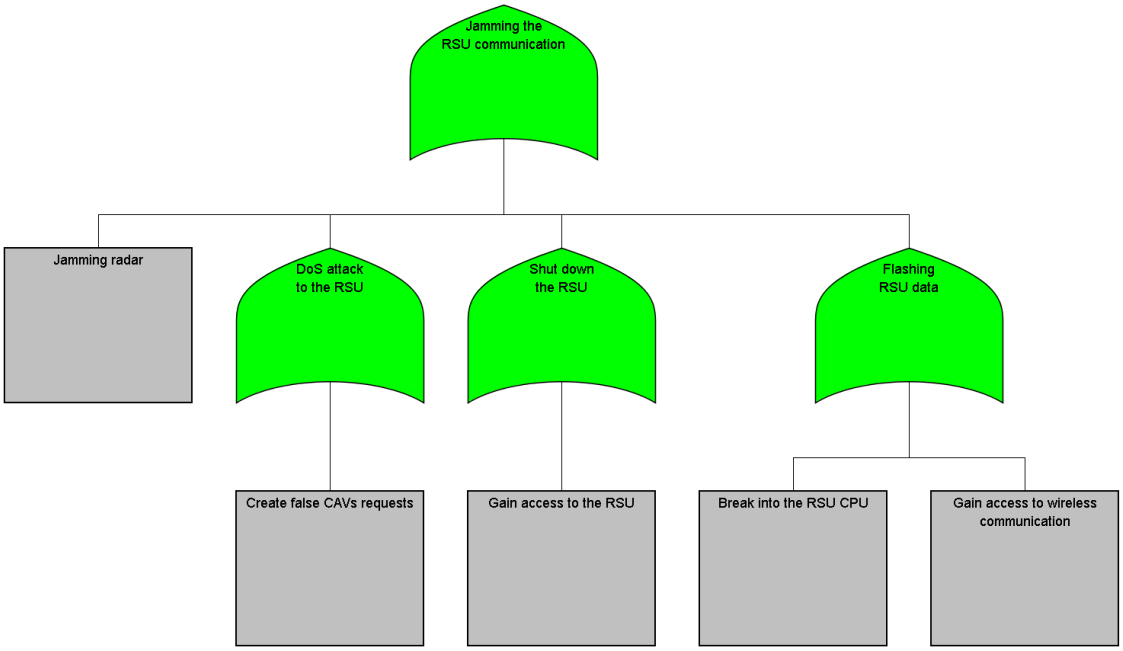

Fig. 40 Jamming the RSU communication.

The next attack tree is named Spoof RSU instructions (Fig. 41), this attack tree is related to the spoofing attack described before but against the RSU. At this attack tree almost the explanation of all the attacks was provided, thus in some cases only the countermeasure is described. The sub attacks regarded are:

- MITIM attack between RSU and CAVs: The countermeasure to this attack is the CAVs authentication and authorization.
- Tampering the RSU: The countermeasure against this attack is a tamper protection for the RSU.
- Remote access to the RSU: The countermeasure against this attack is a tamper protection and well-defined policies in the authorization mechanism.
- MITM attack between LIDAR-RSU: The countermeasures against this attack are the authentication of the LIDAR and the LIDAR data encryption.
- MITM between the CAVs and RSU: Against this attack, the countermeasures proposed are the authentication of the CAVs and the CAVs data encryption.
- Spoofing LIDAR: This attack was regarded earlier and the countermeasures are described in the *BBR data spoofing* attack tree.
- Jamming LIDAR: The countermeasure against this attack is an IDS or IPS control placed in the RSU.
- Spoofing CAV output: The countermeasure against this attack is the CAV authentication.
- Jamming CAV output: For this attack the countermeasure proposed is an IDS or IPS in the RSU and in the CAV.

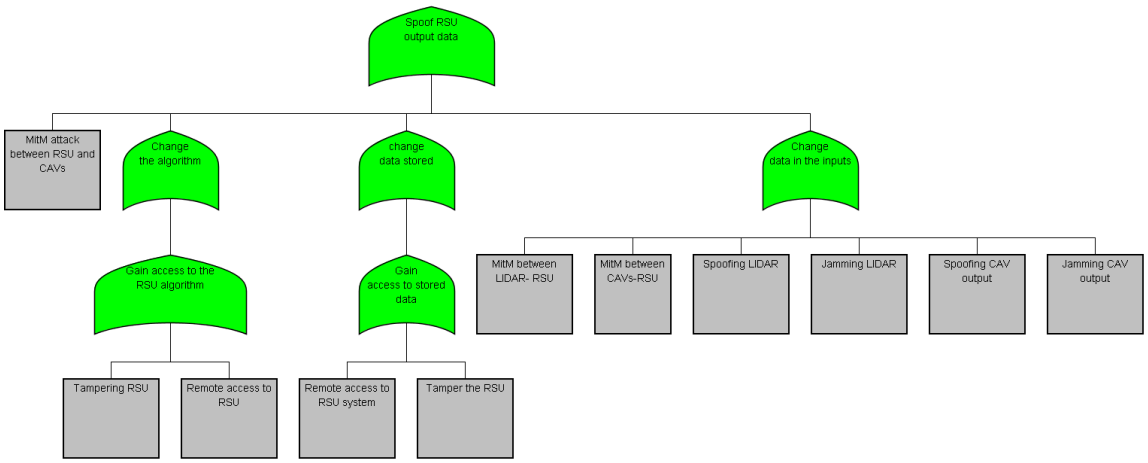

Fig. 41 Spoofing RSU output data attack tree.

The last attack tree regarded for the Flourish security design is the Flashing Control Node data depicted in Fig. 42. The attacks presented below correspond to the leaf attacks, at least one countermeasure is provided for every attack presented below:

- Sniffing data traffic: There were two sub-attacks regarded to reach this attack, they are a back-door creation and MITM attack. The countermeasure provided for these two attacks are IPS or IDS implementation in the processing focusing on the port scanning.
- Provide right credentials to reach remote access: Authentication and authorization mechanisms are needed for any asset that tries to communicate to the processing node.

- Tampering control node: The logical countermeasure against this attack is a tamper protection for hardware and software.

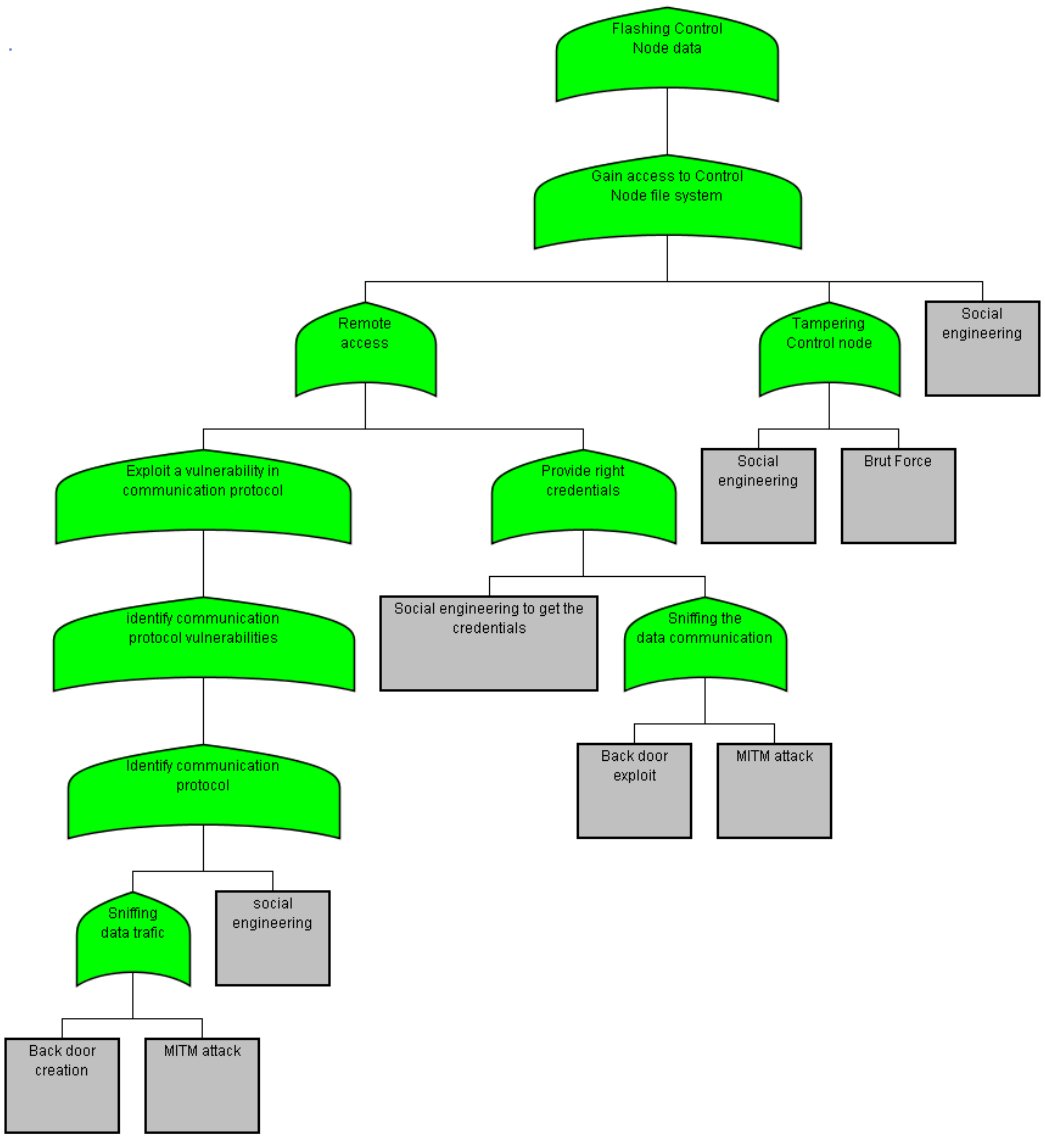

Fig. 42 Flashing Control Node data attack tree.

Once the attack trees were analysed and the countermeasures were identified, it is time to know where they have to be placed. The first approach to the location and identification for the countermeasures is in the countermeasure description presented earlier.

The IoTsecM profile includes some extensions to the use cases metaclasses. The first approach to 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 the use case diagrams for each scenario adding the security countermeasures are provided.

The first scenario described is about the LIDAR and its interactions with the CAV and RSU. The use case diagram related to this scenario is depicted in Fig. 43, for this and the text use cases the only use cases explained are the concerned to the security countermeasures identified. For this scenario the use cases identified are shown in Fig. 43.

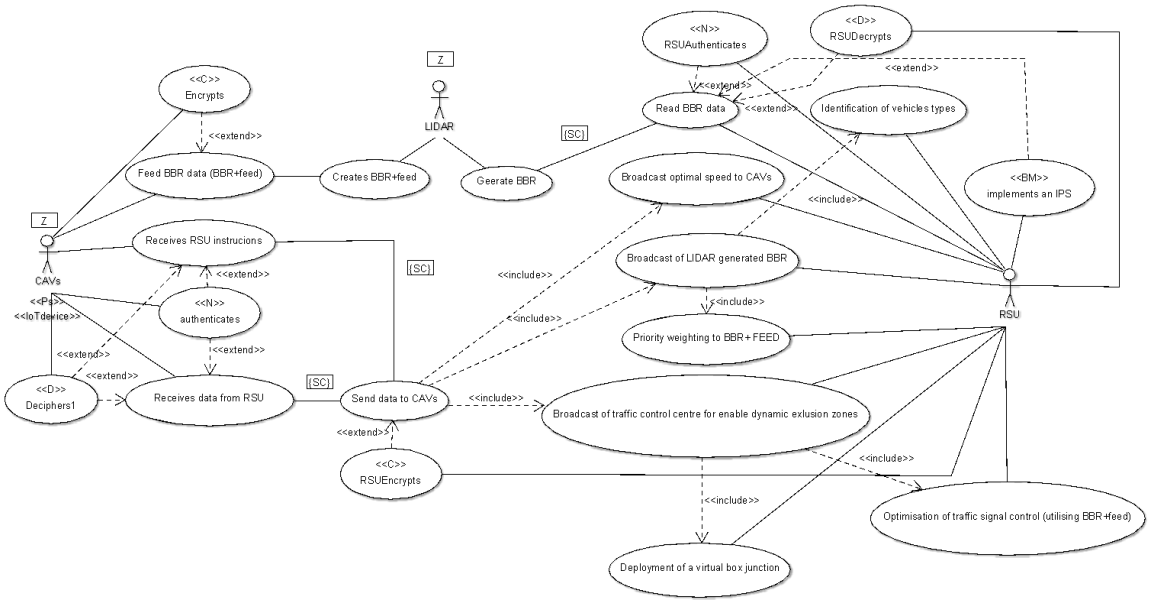

Fig. 43 LIDAR sceneario use case diagram.

The use case concerned to the countermeasures are presented in tables 6 to 20, the tables comprises next fields: Use case name, participating actor, entry condition, flow of events and exit condition.

Table 6 <<C>> use case for CAV, scenario 1.

Use case name: **<<C>> Encrypts**
Participating actor: **CAVs**
Entry condition: **The CAVs receives data from RSU and authenticate and decipher them.**
Flow of events: **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 7 <<N>> authenticates use case for CAV, scenario 1

Use case name: **<<N>> 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 <<N>> element obtains the RSU credentials from the package.** **The <<N>> stereotype instance creates complementary information from de credentials** **The <<N>> stereotype instance runs the authentication function.** **The <<N>> 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 8 <<D>> Deciphers1 use case for CAV, scenario 1.

Use case name: **<<D>> Deciphers1**
Participating actor: **CAVs**
Entry conditions: **True assertion from the <<N>> stereotype use case (*authenticates*).** **The <<D>> stereotype instance holds the CAVs private key according to the pseudonym at the moment.**
Events flow: **The <<D>> Deciphers1 obtains the private key.** **The <<D>> Deciphers1 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 9 <<C>>RSUEncrypts use case for RSU, scenario 1.

Use case name: **<<C>> 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.**

Table 10 <<N>> use case for CAV, scenario 1.

Use case name: **<<N>> 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 <<N>> element obtains the LIDAR credentials from the package.** **The <<N>> stereotype instance create complementary information from de credentials** **The <<N>> stereotype instance runs the authentication function.** **The <<N>> 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 11 <<BM>> implements an IPS use case for RSU, scenario 1.

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

Table 12 <<D>> RSUDecrypyts use case for CAV, scenario 1.

Use case name: **<<D>> RSUDecrypts**
Participating actor: **RSU**
Entry condition: **The RSU reads the BBR data.**
Events flow: **The <<D>> RSUDecrypts obtains the private key.** **The <<D>> 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.**

Besides of the security use cases defined before, there are other constraints displayed on the use case diagram which are placed there to integrate more security concerns within the diagram. The CAVs must be authorized actors; this is depicted with a “Z” over the actor´s head, which means an authorization constraint for all the CAVs actor instances. A pseudonym must be assigned to each CAV, this is depicted with the stereotype <<PS>> applied to the CAVs actor. This means that the security resolution unit provides pseudonyms certificates. It is very common for such certificates to be temporal, hence, the certificates are revoked in a short time, in order to guarantee the privacy of CAVs.

There are two links identified as secure communications constraints. The links are the *Receives data from RSU* and *send data to CAVs* where the communication from RSU to the CAVs is established. The other Secure Communication ({SC}) 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 “N” text box is placed over its head.

The next scenario regarded to collaborative manoeuvring between the CAVs and between CAVs and HDV (Fig. 44). The scenario is not utterly described, nevertheless the uses cases related to the security countermeasures are presented in tables comprises with the fields shown before.

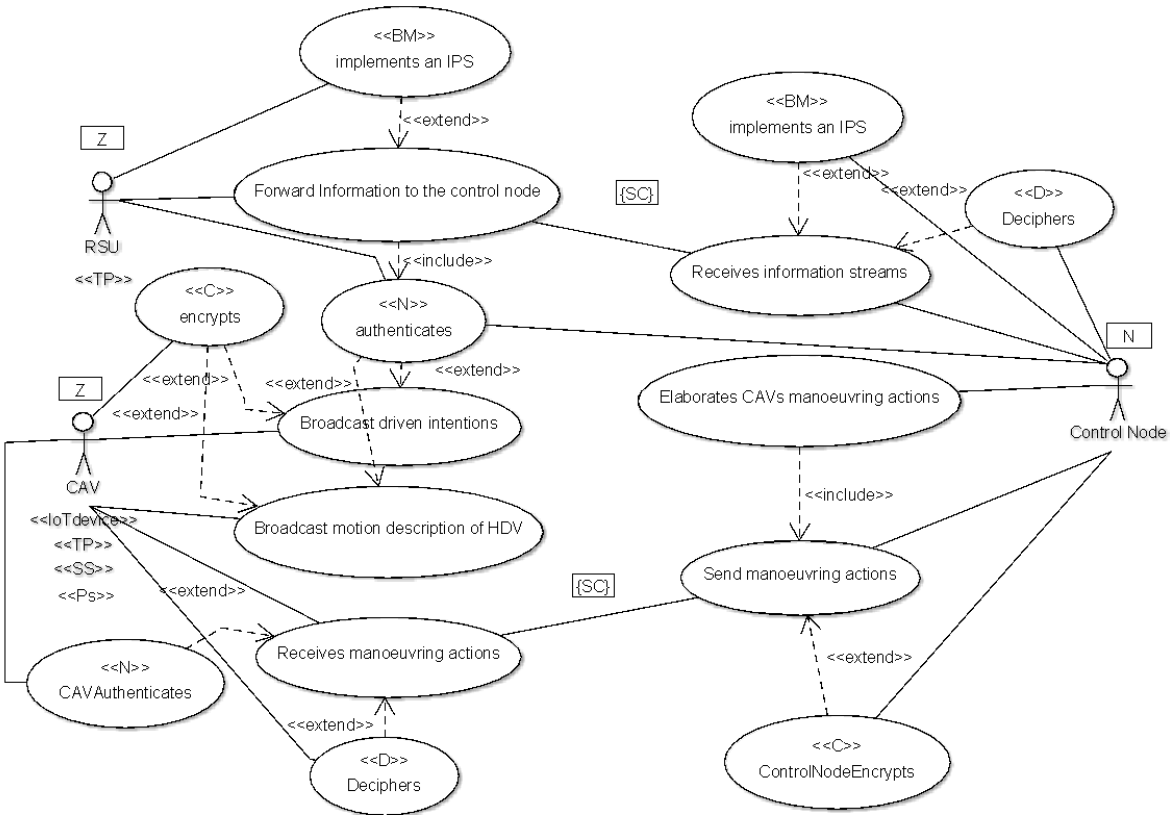

Fig. 44 Collaborative manoeuvring scenario use case diagram.

Table 13 <<N>>authenticates use case for RSU, scenario 2.

Use case name: **<<N>> authenticates**
Participating actor: **RSU**
Entry condition: **The RSU receives the CAV driven intentions and the HDV motion description**
Events flow: **The package is received.** **The CAVs actor runs the authentication element.** **The <<N>> element obtains the CAV credentials from the package.**

The <<N>> stereotype instance create complementary information from de credentials **The <<N>> stereotype instance runs the authentication function.** **The <<N>> creates the assertion {True, False}.** **This use case extends the *Broadcast driven intensions* use case and *Broadcast motion description of HDV* use case**
Exit condition: **The RSU authenticates the CAV which broadcast the data**

Table 14 <<BM>>authenticates use case for RSU, scenario 2.

Use case name: **<<BM>> implements an IPS**
Participating actor: **RSU**
Entry condition: **Receive data from the CAV.** **IPS preconfigured.**
Events flow: **The RSU calls the <<BM>> Implements a preconfigured IPS.** **According to the intrusion detected the IPS reacts.** **The IPS upgrade the policies in the <<Z>> element.** **The <<BM>> implements an IPS use case extends the forward information to the control node**
Exit condition: **The <<BM>> instance is running rightly.** **The IPS is monitoring.**

Table 15 <<C>>encrypts use case for CAV, scenario 2.

Use case name: **<<C>> encrypts**
Participating actor:

CAV
Entry condition: **The CAV receives data from the on-board sensors.**
Flow of events: **The CAV creates its driven intentions data payload.** **The CAV creates the motion description of HDV data payload.** **The <<C>> instance retrieves the control node public key.** **Encrypts the data using the control node public key.** **The CAV signs the package using its temporal certificate.** **This use case extends the *Broadcast driven intentions* use case.** **This use case extends the *Broadcast motion description of HDV* use case.**
Exit condition: **The data package is signed and sent by the CAVs to the RSU**

Table 16 <<N>>CAVAuthenticates use case for CAV, scenario 2.

Use case name: **<<N>> CAVAuthenticates**
Participating actor: **CAV**
Entry condition: **An entry package is sent from the control node containing the manoeuvring data.**
Events flow: **The package is received.** **The CAV actor runs the authentication element.** **The <<N>> element obtains the control node credentials from the package.** **The <<N>> stereotype instance creates complementary information from de credentials** **The <<N>> stereotype instance runs the authentication function.** **The <<N>> creates the assertion {True, False}.** **This use case extends the *Receives manoeuvring actions* use case.**

Exit condition: **The CAVs authenticate the package received**

Table 17<<D>>Deciphers use case for CAV, scenario 2.

Use case name: **<<D>> Deciphers**
Participating actor: **CAV**
Entry conditions: **True assertion from the <<N>> stereotype use case (*authenticates*).** **The <<D>> stereotype instance holds the CAVs private key according to the pseudonym at the moment.**
Events flow: **The <<D>> Deciphers obtains the private key.** **The <<D>> Deciphers apply the decryption algorithm.** **The data from the control node is in plain text now for the CAVs interpretation.**
Exit condition: **The data decryption was correctly done.**

Table 18 <<C>>ControlNodeEncrypts use case for Control Node, scenario 2.

Use case name: **<<C>> ControlNodeEncrypts**
Participating actor: **Control Node**
Entry condition: **The control node already calculated CAVs manoeuvring actions**
Flow of events: **The control node creates CAVs manoeuvring data payload.**

The <<C>> instance retrieves the corresponding CAVs public keys. **Encrypts the data using the CAVs public keys.** **The control node signs the package.** **This use case extends the *Send manoeuvring* use case.**
Exit condition: **The data package is signed and send by the control node to the CAVs**

Table 19 <<D>>Deciphers use case for Control Node, scenario 2.

Use case name: **<<D>> Deciphers**
Participating actor: **Control Node**
Entry conditions: **True assertion from the <<N>> stereotype use case (*authenticates*).** **The <<D>> stereotype instance holds the control node private key.**
Events flow: **The <<D>> Deciphers obtains the private key.** **The <<D>> Deciphers apply the decryption algorithm.** **The data from the CAVs about their motion description of HDV and the CAVs manoeuvring intentions is in plain text now for the control node interpretation.**
Exit condition: **The data decryption was correctly done.**

Table 20 <<BM>>Control node use case for Control Node, scenario 2.

Use case name: **<<BM>> implements an IPS**
Participating actor:

Control node
Entry condition: **Receive information streams from CAVs forwarded by the RSUs.** **IPS preconfigured.**
Events flow: **The Control Node calls the <<BM>> Implements an IPS.** **According to the intrusion detected the IPS reacts.** **The IPS upgrade the policies in the <<Z>> element.** **The <<BM>> implements an IPS use case extends the *Receives information stream* use case**
Exit condition: **The <<BM>> instance is running rightly.** **The IPS is monitoring.**

The last constraints related to this scenario are depicted on the use case diagram as well, they are the authorization requirement for CAVs, the authorization requirement for the RSU and the authentication requirement for the Control Node. The communication links with the {SC} constraint added on them are the links that connect the Control Node and CAV where deuce the *Send manoeuvring actions* and *Receives manoeuvring actions* use cases are linked. There are new requirements for the CAV and the RSU, for the CAV a <<TP>> stereotype is needed in order to request a tamper protection for the hardware and software within the CAV besides of the <<SS>> stereotype which indicates that a secure storage is needed there as well. For the RSU a <<TP>> stereotype is depicted as well.

The next scenario analysed is related to especial zones assets, which involves the Carer, RSU and CAVs actors. The use case diagrams for this scenario is shown in Fig. 45. As in previous scenarios each use case that targets the security countermeasures identified is presented in a table.

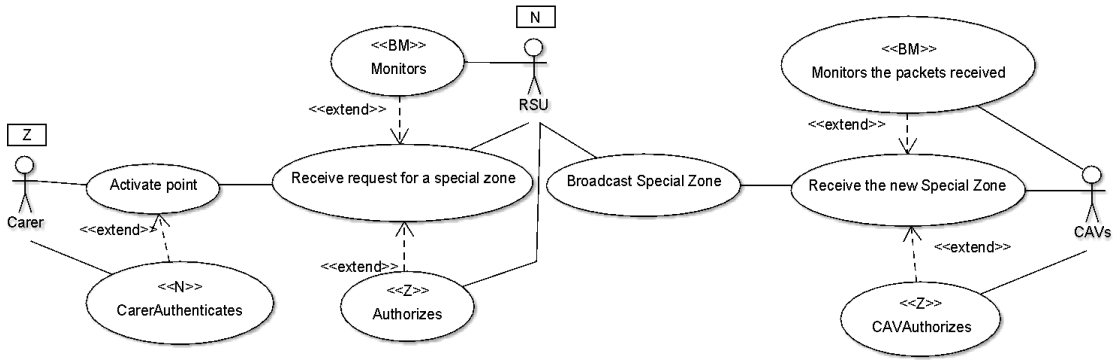

Fig. 45 Special zone request scenario use case diagram.

Table 21 <<N>>CarerAuthenticates use case for Carer, scenario 3.

Use case name: **<<N>> CarerAuthenticates**
Participating actor: **Carer**
Entry condition: **An activation point is previously installed in place.**
Events flow: **The carer verifies the activation point.** **The carer types its password on it.** **This use case extends the *activate point* use case**
Exit condition: **The carer authenticates itself in the activation point.**

Table 22 <<Z>>RSUAuthorizes use case for RSU, scenario 3

Use case name: **<<Z>> RSUAuthorizes**
Participating actor: **RSU**

Entry condition: **A request was sent from the activation point for a special zone.**
Events flow: **The RSU verifies the activation point authenticity.** **The <<Z>> instance authorize the activation point requester to communicate to the CAVs.** **The <<Z>> Authorizes use case extends the *Receive request for a special zone.***
Exit condition: **The RSU authorises the activation point.**

Table 23 <<BM>> monitors the packets received use case for CAV, scenario 3.

Use case name: **<<BM>> monitors the packets received**
Participating actor: **CAV**
Entry condition: **Receive request for a special zone.** **IPS preconfigured.**
Events flow: **The CAV calls the <<BM>> Monitors the packet received.** **According to the intrusion detected the IPS reacts.** **The IPS upgrade the policies in the <<Z>> element.** **The <<BM>> implements an IPS use case extends the *Receives information stream* use case**
Exit condition: **The <<BM>> instance is running rightly.** **The IPS is monitoring.**

Table 24 <<Z>>CAVAuthorizes use case for CAV,, scenario 3.

Use case name: **<<Z>> CAVAuthorizes**
Participating actor: **CAV**
Entry condition: **A request was sent from the activation point for a special zone.**
Events flow: **The CAV verifies the activation point authenticity.** **The <<Z>> instance authorizes the activation point according to the policies within the ACL.** **The CAV activate the special zone, and apply the instructions received.**
Exit condition: **The CAV authorises the activation point and the special zone request.**

The Carer, in this scenario, is an authorized actor, since it must provide the right credentials in order to activate the special zone request. This means that it has been authorized by Flourish to be a carer. The way how IoTsecM depicts it, is placing a Z over its head to indicate that it requires to be an authorized actor.

The next scenario regarded is related to the unplanned incident management, in this case, with the previously presented security use cases is enough to ensure this scenario. The CAV needs to be an authorized actor, the RSU holds a <<Z>> authorizes module in order to guarantee that the CAV sending data is included in the ACL. The use case diagram for this scenario is shown in Fig. 46.

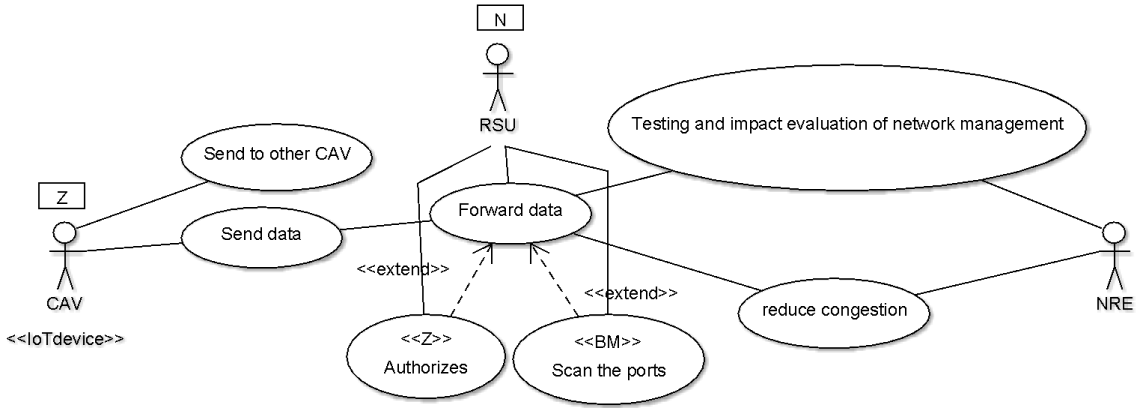

Fig. 46 NRE unplanned incident management scenario use case diagram.

The next scenario is depicted in theFig. 47 involves the NRE and Network AI unit assets, the security countermeasures for the RSU and CAVs have been already introduced. Therefore, only the countermeasures for the NRE and Network AI unit are presented.

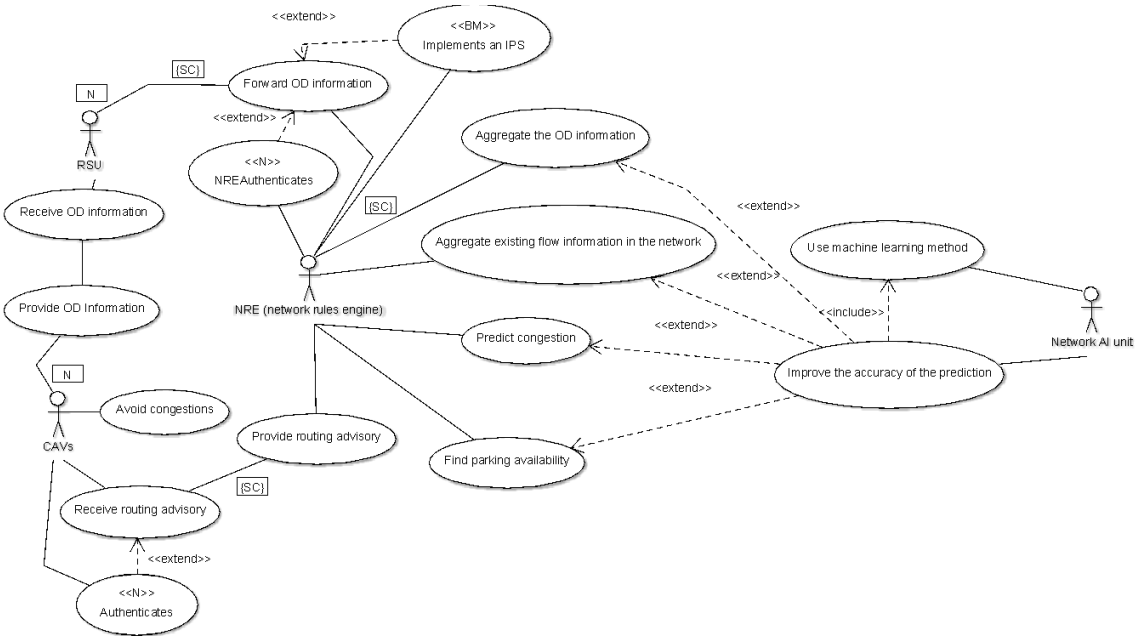

Fig. 47 Parking advisory scenario use case diagram.

Table 25 <<N>>NREAuthenticates use case for NRE, scenario 4.

Use case name: **<<N>> NREAuthenticates**
Participating actor: **NRE**
Entry condition: **An entry package is sent from the CAV passing through the RSU.**
Events flow: **The package is received.** **The NRE actor runs its authentication element.** **The <<N>> element obtains the control node credentials from the package.** **The <<N>> stereotype instance creates complementary information from de credentials** **The <<N>> stereotype instance runs the authentication function.** **The <<N>> creates the assertion {True, False}.** **This use case extends the *Forward OD information* use case.**
Exit condition: **The CAVs authenticate the package received**

Table 26 <<BM>>implements an IPS use case for CAV, scenario 4.

Use case name: **<<BM>> implements an IPS**
Participating actor: **CAV**
Entry condition: **Receive OD information package data.** **IPS preconfigured.**
Events flow: **The NRE calls the <<BM>> implements an IPS.** **According to the intrusion detected the IPS reacts.** **The IPS upgrade the policies in the <<Z>> element.**

The <<BM>> implements an IPS use case extends the *Forward OD information* use case
Exit condition: **The <<BM>> instance is running rightly.** **The IPS is monitoring.**

The analysis of the countermeasures identified allows to know the place where the security mechanism should be. The use cases diagrams resulted very useful for the requirements conceptualization, thus the IoTsecM extensions within the use case diagrams could depict each security countermeasure identified in the attack trees.

The next step is to propose the whole system architecture, here the functional elements and the non-functional elements are shown in a class diagram. This helps to attend all the issues concerning to the interconnections between the assets, the identification of their operations and the relation between the security mechanisms.

The IoTsecM profile includes extension for classes, components and devices metaclasses, which assist the designing of the system architecture. In Fig. 48 the system architecture regarding the security elements is depicted. The objective of the IoTsecM profile is to allow the designers to build, model and depict the security mechanisms together with the functional elements, in this a complete landscape for the system may be conceptualised and then, the system architecture can be reached.

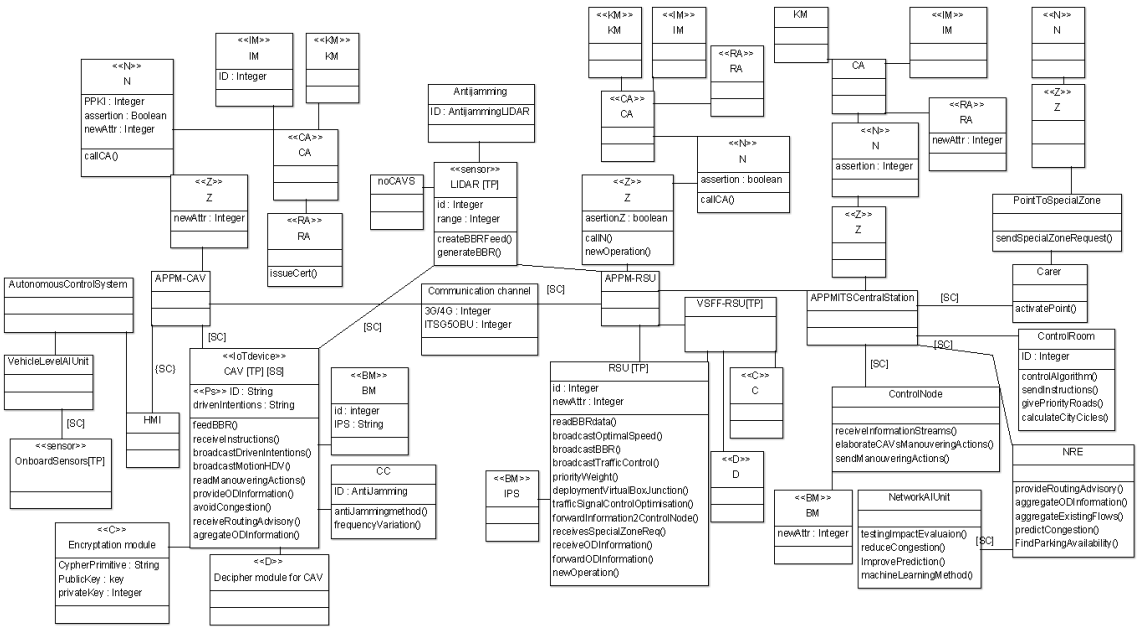

Fig. 48 Flourish architecture applying the IoTsecM profile.

As it can be seen in the Fig. 48, the security countermeasures identified before are depicted in the system architecture, the CAV requires a tamper protection and secure storage, besides it requires a pseudonym. As in the use cases was shown the CAV authenticates, authorizes, encrypt and decrypt besides it monitors the entry data and the network hence, the <<N>>, <<C>>, <<D>>, <<BM>> and <<Z>> stereotypes are instanced.

The addition of other security elements such as CA, RA, IM and KM is because of the necessity of certificated to be issued and the pseudonyms requirements, all this element conform a PKI, or in the case of the pseudonyms requirement, it is named pseudonym public key infrastructure (PPKI).

The RSU must contain the security countermeasures found. Therefore, the stereotypes instanced associated to the RSU are <<N>>, <<C>>, <<D>>, <<BM>> and <<Z>>, besides as well as the CAV the PPKI infrastructure is supported by the RSU. A 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 modelled contained in a central station, the central station contains the stereotype instances <<N>>, <<C>>, <<D>>, <<BM>> and <<Z>>.

The IoTsecM proposal allowed to depict the security concerns applying the stereotypes described before. Once threat analysis was performance, 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 needs to be placed, which actor is associated to 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 are depicted in order to observe the processes followed by them, besides of their interaction.

5.2 IoTsecM designing security in an mHealth application

The IoT can be characterised by interworking networks which incorporate physical objects. These objects have virtual representations and are capturing data corresponding to real physical characteristics obtained and observed by sensors. There can be many relationships between the things in an IoT systems, even without human interaction. The IoT uses the current Internet as a communication medium, besides the IoT environment may involve other domains, for instance in many IoT systems a mobile Application (mApp) or web Application (webApp) allows the user to communicate with the physical entity through a smart phone app. Therefore, the IoT and mApps may coexist within a particular system.

The IoTsecM approach allows the depicting and modelling of the security concerns from the analysis stage within a waterfall development life cycle. In this process the security requirements are depicted together with the functional requirements in order to improve the conceptualisation of the system security requirements, the relationship with the system assets and the possible issues of adding security mechanisms. The consideration of security requirements into the analysis stage eventually will change the system architecture, since the aggregation of security may add costs in the designing stage, but it also may decrease the security response costs in case of a security incident.

The study case regarded for this section is named Dentify.me app, it is part of the mHealth domain. The mHealth domain is one of the emerging healthcare delivery models of eHealth which utilises recent technologies in the delivery of health treatment to enhance collaboration, communication and coordination in the health sector. Therefore, mHealth supports health by incorporating mobile devices into the healthcare delivery model; mHealth involves the use of many services and utilities supported by a mobile phone device such as. Consequently, mHealth as an application of IoT systems is expected to grow in the next years, this will have a massive impact on the way healthcare is delivered in our modern world as it enables affordable personal management of the user's well-being. The analysis and design of the IoT system related to mHealth must consider the security requirements, since the information that mHealth collects is very sensitive and it would imply a huge risk if it falls in the wrong hands, for example due a security attack.

In particular, mHealth has played an important role in emergency aid in the event of a disaster where it enables victims to record personal medical information. This, has facilitated the rescue team to give the victim the right treatment based on informed-decisions. Therefore, mHealth implies privacy concerns for the users, since the information collected is related to medical data which are confidential, and they can be very dangerous on the wrong hands.

Tipically, mHealth systems use GPS as a vital asset to detect the disaster's location in global disaster responses. With the aid of GPS the rescue team can locate the victims in some cases almost in real time, this makes the rescue faster, easier and more accurate.

The study case regarded for this section is named Dentify.me app, which is a mHealth app that helps to recue living victims, find the missing ones, and identify the found victims in natural disasters or in horrifying mass fatality incidents. The victims need to be located or reported to be provided with first-aid paramedic services. All the victims need to be identified and listed in order that all Disaster Victim Identification (DVI) efforts can be best exploited to find the missing victims. The deceased victims need to be identified. Rescue teams rely on victims or eyewitnesses to report incidents manually or on post-active systems for receiving emergency calls.

Dentify.me App proposes a victim-centred solution that automates the process of requesting S.O.S. and locating victims, generating missing persons list and collecting and delivering pre-disaster data. They use semi-structured interviews in order to identify and gain good understanding of the

pre-incident and post-incident data needs, the teams involved, information shared and information security needs and challenges. Dentify.me App engages with eyewitnesses to automate the incident detection and identification process. Dentify.me App uses hard systems methodology to automatically identify and locate potentially affected victims, alert the rescue team to reach out for the people alive, generate a list of missing ones, and collect and deliver pre-incident data to the DVI team.

The Dentify.Me App team fully designed, implemented and tested the proposed mobile health solution, nevertheless the security analysis for the system integrity, confidentiality and availability was not realized. Therefore, a well-motivated attacker may be colluded with malicious people to injure or even kill people; the informatics attacker may attack the system infrastructure and assets, provoking the system to do not respond or respond incorrectly. It would be even more catastrophic since the rescue teams would not be able to react at the right time and in the right way and as a consequence would be deader and injured people. The security analysis must be done, in particular, in systems related to eHealth and in particular in mHealt systems.

In this work, the security analysis of Dentify.Me App was done, and the process followed is the same as the presented in the last section. The process followed to perform the threat modelling and security countermeasures analysis and design is based on [49]; however, this process was customised and extended in order to add the countermeasures modelling, the process followed is summarised into the next steps:

- Identify the assets
- Create an IoT system architecture overview
- Decompose the IoT system
- Identify threats
- Document threats
- Propose counter-measures for each threat
- Propose a system architecture depicting security countermeasures

First, a clearer idea of the system is needed. As it was mentioned earlier, the study case is the Dentify.me App which has been already designed, nevertheless the security requirements had not been regarded, so the security analysis needs to be performed. As it was mentioned in the steps

introduced before, the assets identification within the system offers an understanding of what needs to be protected, it may include: people, hardware, software, procedures, data information, etc. It is an inventory of all the items (virtual or physical) that are important for the system or for the organisations and can be of the attacker's interest. In the Dentify.me App an assets classification is provided in order to identify the assets according to its particular type:

- People
    - Eyewitness
    - Injured and Missing victims
    - Rescue team
    - DVI team
- Hardware
    - Mobile
    - Wearable devices
    - Desktop
    - Printer
    - Device for recognition
    - Switch
    - Router
    - Cable for connection
- Software
    - Firewall software
    - Operating system software, Network operating system
- Procedure
    - User registration in app
    - Basic information from all users
    - Identification and location of victims
        - List of potential missing people
        - Request a response from every victim on that list
        - Starts timing and generates four sub-lists:
            - Sub-list: Alive and Well Victims
            - Sub-list: Alive and injured Victims
            - The list shared with the rescue team

        - Sub-list: Pending Confirmation
        - The list shared with DVI team
        - Create list for missing people sub-list
    - App allow the victim to trigger S.O.S request
    - Rescue team is able to access Affected Victims List
- Data/Information
    - Basic information from all user
    - Data (Pictures/Video/Notes)
    - Information
- Networking
    - Server
    - Host
    - Clients

Once the assets are identified they are placed together in an architectural view in order to observe their associations; the Dentify.me App architecture can be find in [73]. The class diagram is used as a conceptual diagram in order to show how the assets are located within the system and to observe how they are associated. In Fig. 49 the architectural view is shown.

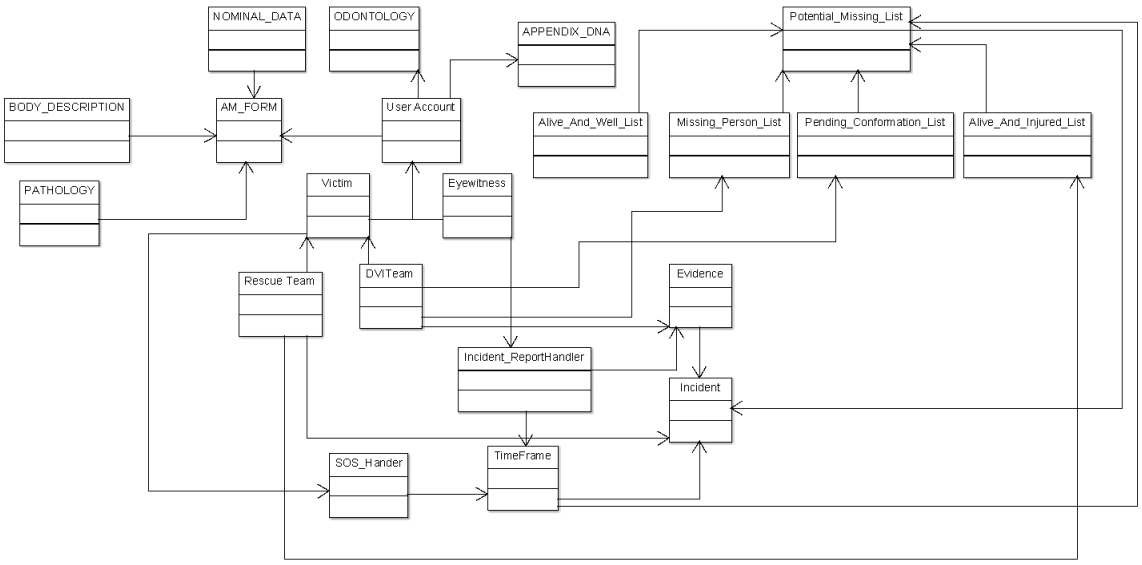

Fig. 49 Dentify.Me App architectural view.

At this point the system architecture was depicted and analysed, the assets identification was done, and hence it is time to analyse the security requirements. As it was mentioned the attack trees are an orderly and sequential way of describing the sub-attacks to compromise a system, they are a useful tool to conceptualise and visualise the possible attacks, where the designer must put himself in the attacker's shoes to devise all the different ways in how an asset can be compromised. This analysis results in the underlying root causes of attacks, allowing to create attacker profiles, in order to 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, as far as possible, the entire attack surface. It is recommendable to do it within a work group where at least two people can build it together. The tool used for this purpose is the same that in the previous study case, the tool we are using is named SecureItree, an attack tree modelling tool built by the Canadian company Amenaza (the Spanish word for threat)) In this tool the root node represents the end objective and the children nodes depict the different sub attacks in order to accomplish the overarching goal. The nodes can be AND operator, OR operator, or a LEAF. The AND operator means that all of the children nodes are needed to accomplish the parent node. On the other hand, the OR operator means that any of the children nodes satisfy the parent node.

The first attack tree to analyse is the one with the parent node “Eyewitness unable to report Incident” shown in Fig. 50, meaning that the witness is not able to report an incident because some or some sequence of attacks is being executed by an attacker. The attacks related to social engineering are particularly difficult to address since they rely on the human mistake, the countermeasures against that kind of attack is to educate the people to know the possible ways in how an attacker may achieve their goal, for instance the attackers may provide right credentials, and hence stronger policies for physicals human access control need to be enforced, nevertheless that kind of attacks is out of the scope of IoTsecM.

The attacks and sub attacks to the Dentify.me App assets are described in this section; however, not all the attacks are described, since some of them are not a relevant part of the attack vector. Therefore, the attacks description provided below involves the relevant attacks accompanied with the corresponding countermeasure. The nodes that are the offspring of “Eyewitness unable to report Incident” threat (Fig. 50) are:

- Error in data entry: The sub attacks describe below attempts to provoke an error in the data entry. To mitigate this attack the sub attacks needs to be mitigated and as a consequence of that this attack will be mitigated as well.
    - Malicious app changes the data
        - Download and Install malicious app
            - Phishing attack: With malicious links and another technique the attackers may force the users to install malicious apps in their smart phone, in this attack those malicious apps provoke data changing and as a consequence errors in data entries. The countermeasure against this attack is to have an anti-virus installed within the smart phone which prevents the malicious apps installation.
            - Colluded app: An app is installed which is colluded with other app that provokes the error in data entry. The countermeasure against this attack is to encapsulate (sand boxing) the *Dentify.Me App*, in order to deny the access to any other app within the smart phone

to the Dentify.Me App. Therefore, the countermeasure is an authorisation process between the smartphone apps.

- Error in the app:
    - The app was not loaded
        - Malicious App Installed: The malicious app do not let the *Dentify.Me App* to load, hence, the eyewitness is unable to report incident. The countermeasure against this attack is an authorisation process between the smartphone apps.
- No Internet connection
    - Wifi jamming
        - False wifi Access Point (AP): The attacker creates a false access point, then the data sent from the smart phone cannot be received by the *Incident_ReportHandler* or it may be modified. This kind of attacks are part of the MITM attacks. The countermeasure against this attack is to provide an authentication method between the eyewitness and the *Incident_ReportHandler*.
    - No mobile data
        - No cellular network
        - Cellular Network Jammer Attack: This kind of attack is very common to compromise a wireless environment such as the communication between the eyewitness and the *Incident_ReportHandler*, the goal is to drop the signal to a level where the communication is interrupted. Typically older wireless area networks are the most vulnerable to the success of this kind of attack, since current networks are able to adapt to unintentional or intentional interference. The countermeasure proposed for this attack is an intrusion prevention system (IPS), since it should be able to detect the presence of any unauthorised client device.
- Camera Malfunction:
    - Malicious App disable the camera
        - Download and Install malicious app: These attacks involve the installation of malicious apps, they were previously described, as well as the

countermeasures proposed. The only change is that these attacks attempts to provoke a camera malfunction.

- Phishing attack
- Colluded Apps

- Break camera

- Mobile battery discharged:
  - Malicious app changes the battery level.
- Steal the Eyewitness mobile phone:
  - Direct attack to the eyewitness mobile: This is a physical attack; the attacker steals the eyewitness mobile phone.
- Server does not respond:
  - DoS attack to the server: This attack consists in sending many requests to the server, in order to make it attend just the false and mal formed requests whereas they deny any other request even the authenticated requests. The countermeasures against this attack is an intrusion detection system (IDS) or IPS.

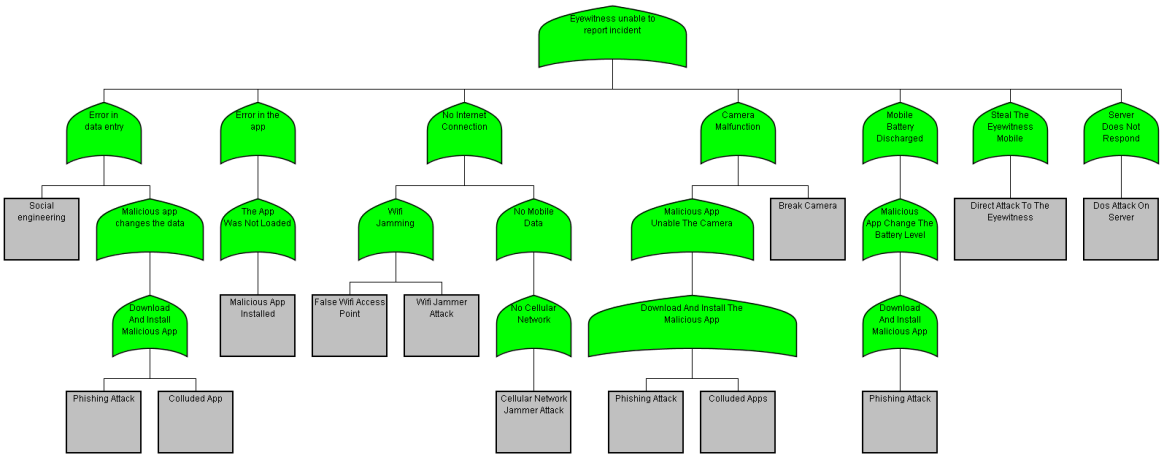

Fig. 50 "Eyewitness unable to report Incident" attack tree.

The second threat analysed is the "Rescue Team cannot access affected Victim List" this means that the rescue team has not access to the affected victim list, it is displayed Fig. 51 and it considers the next possibilities or sub attacks:

- No Internet connection
    - False Access Point: A MITM attack is performed by an attacker, the attacker creates a false access point and therefore is able to receive, modify or block the communication between the recue team and the affected victim list. The countermeasure proposed against this attack is the authentication of the rescue team, and a trusted victim list.
    - Wifi jammer: The countermeasure against this attack is an IDS or IPS control.
- Access a false affected Victim List
    - Modify, delete, observe the list
        - Password force Brute-Attack: The attacker performs an attack against the server access mechanism by a brute-force attack, this means that according to a password dictionary the attacker tries each one of the possible combinations. The countermeasure against this attack is a well-defined authorisation mechanism and an IPS in the server.
        - Social engineering.
- Denial of Service attack: The countermeasure against this attack is an IPS placed in the server, which is able to monitors the port server.

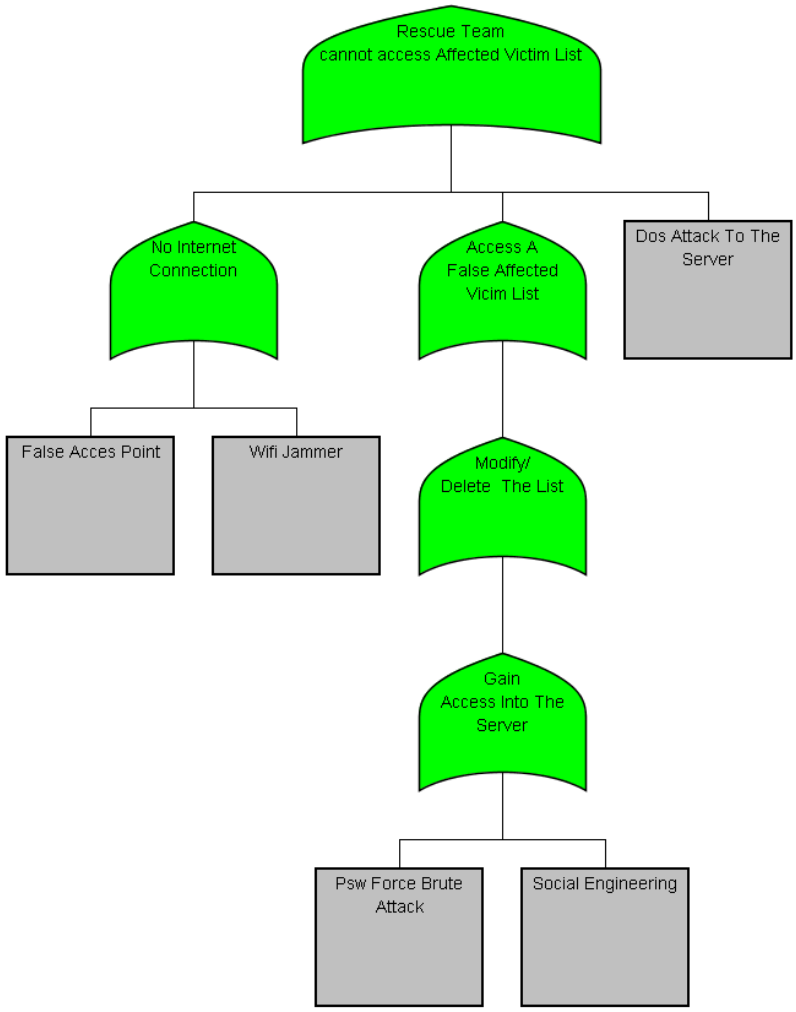

Fig. 51 Rescue Team cannot access affected victim list attack tree.

The next attack tree has as a root node “Communication interception from mobile”, this means that an attacker aims to eavesdrop the data traffic from the mobile to the server, the attack tree regarded for this threat is shown in Fig. 52:

- Communication interception from mobile
- Man in the middle (MITM) attack: The attacker performs a MITM attack in order to intercept the communication, this can be performed in several ways, however the countermeasure proposed against this attack are authentication and authorisation controls from the mobile to the servers.
- Back door attack to the server: The attacker identifies a vulnerable service, and perform a back door attack in order to take control of the server, in this case taking control of the server

communications. The countermeasure against this attack is an IPS control in the server and authentication mechanisms in each service for the mobiles.

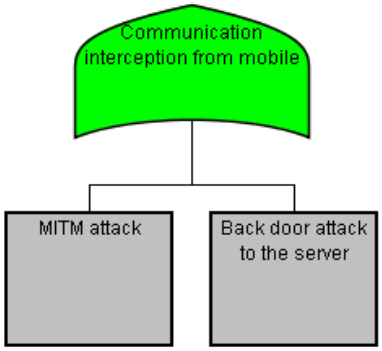

Fig. 52 Communication interception from mobile attack tree.

The next attack tree is “Database theft of basic from all users” where an attacker tries to modify, delete or observe the data base, this would imply an unauthorised access to the DB. The attack tree is shown in Fig. 53, and deployed:

- Database theft of basic information from all users.
- Unauthorised access to the DB
- Password brute force attack: This is an attack against the server access mechanism by a brute-force attack, this means that according to a password dictionary the attacker tries each one of the possible combinations to gain access to the database. The countermeasure against this attack is a well-defined authorization mechanism and an IPS in the server in order to identify and react against the malicious behaviour.
- Social engineering.
- SQL injection attack: This attack is performed entering crafted data that provokes the input to be interpreted as part of SQL query instead of data. The countermeasure against this attack is the sanitisation and validation that could be part of an authorisation mechanism. The objective is to ensure that any malicious characters are not passed to a SQL query for data.

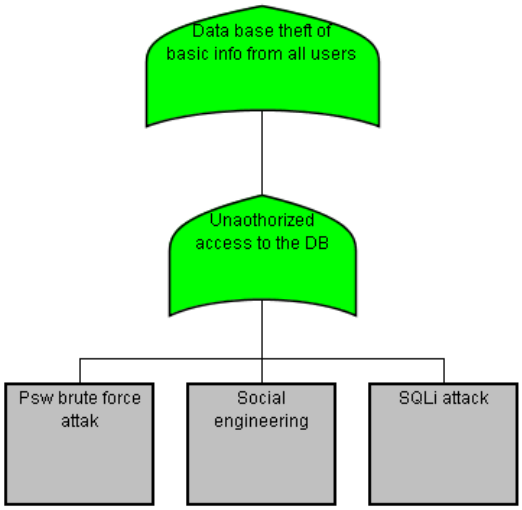

Fig. 53 Database theft of basic information from all users attack tree.

The next case considered is “Recognition device does not recognize”, this means that the mobile camera or any other device dedicated to recognising does not work properly, this could be because of a mobile malfunction or because of an attack and therefore, the attack tree for that threat was modelled and depicted in Fig. 54. The countermeasure against the two attacks are described:

- Wifi jammer: The countermeasure against this attack is an IDS or IPS control. However, as it was mentioned, the recent Wifi AP contain anti jamming technology, thus, another recommendation is to do not use old Wifi AP.
- False AP: The countermeasure against this attack is an authentication mechanism in order to verify the identity of the recognition device.

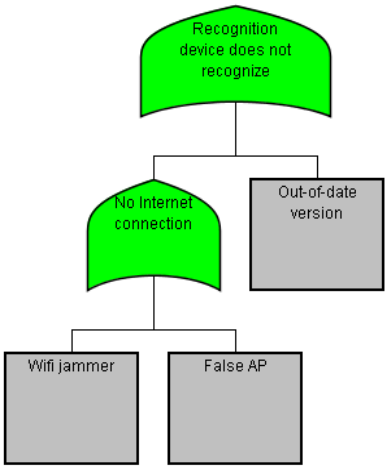

Fig. 54 Recognition device does not recognize attack tree.

The analysis of the "Unauthorised access to router" threat (Fig. 55) is when an attacker crosses the router because none security mechanism is implemented the children nodes that the analysis gave as results are:

- Unauthorised access to router
- Old equipment
- Attack software or Hardware vulnerability: The countermeasure against this attack is to place only current equipment for the *Dentify.Me App* system.
- None access control mechanism implemented: The countermeasure against this attack is an authorisation mechanism implemented in the router.

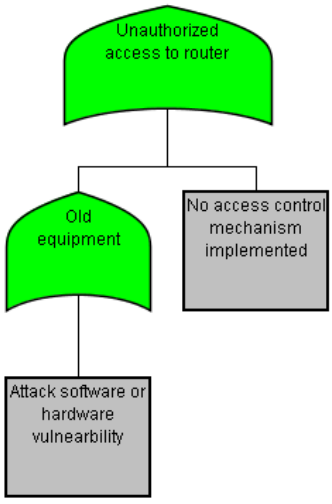

Fig. 55 Unauthorised access to router attack tree.

The next attack tree considers the "Unauthorised access to the lists" (Fig. 56), this means that an attacker will focus his efforts to get in the database server with the aim of modify, observe or delete the *Dentify.Me App*. The sub-attacks regarded are:

- Unauthorised access to the lists
    - Social engineering
    - Get in with right credentials
        - Password brute force attack: This attack was already described; however, the countermeasure against this attack is a blend of an authorisation mechanism and an IPS control.
        - Social engineering.
    - Physical penetration
        - Social engineering
- Credentials falsification

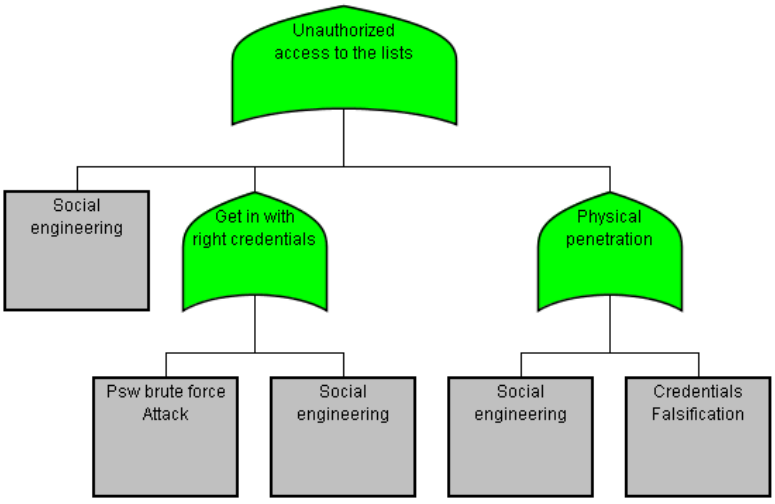

Fig. 56 Unauthorised access to the list attack tree.

The non-repudiation security requirement is a sub-branch of the general availability requirement. In this case an authorised actor needs to be able to access the resources that he requests. In the case of *Dentify.Me App* the name of the attack against this security requirement is "DVI team cannot access a Pending Confirmation List" the analysis results are shown below (Fig. 57). The countermeasures against these attacks were described before, hence they are not described in this section, and only the sub-attacks names are shown:

- DVI cannot access the Pending Confirmation List
- No Internet connection
- False access point
- Wi-Fi jammer
- Logic error programming
- Error in data entry
- DoS attack server

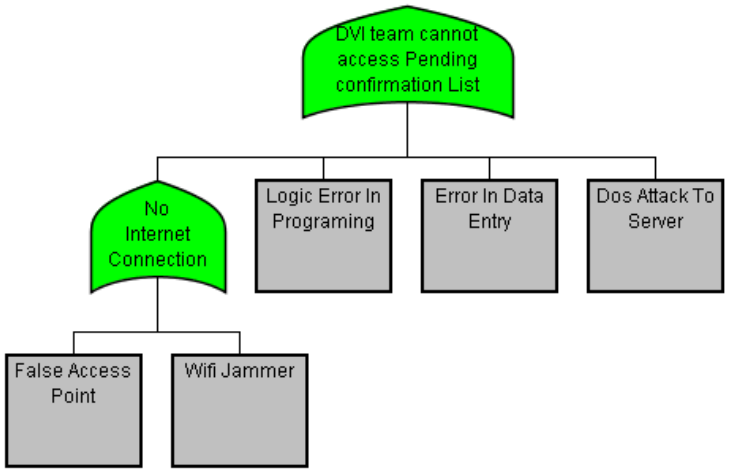

Fig. 57 DVI team cannot access pending conformation list attack tree.

Some attacks are solved with user´s good practices; others need to be taken care of during the programming of the application such as social engineering attacks that are usually accomplished in 3 ways:

- Persuading the user: In the case of Dentify.me app users, it is when the user downloads a malicious application that blocks the camera, for instance.
- Get in physically into the servers: With fake credentials or convincing people, are the most common practices.
- Remote access: With the correct passwords obtained indirectly or directly from users, programmers, etc.

Therefore, users should be made aware to not install applications that appear malicious. For example, applications that ask for permissions that are not congruent with the application intended performance. If this recommendation is followed, then the risk of having an application that blocks a hardware asset will be greatly reduced assuring the correct functioning of *Dentify.Me App*, especially in urgent situations.

Another form of attack is phishing where a malicious link is given to download some software. This can be done through fake emails or web pages that contain them, therefore, users and members of the DVI team and Rescue team are advised not to download anything out of necessary and validated applications.

For the case of the Jammers, which are used when the attacker intends to gain time for a physical attack, e.g., it can be used to deny the opponent the opportunity to communicate in time in a critical situation. The countermeasure for this attack could be very complex, in the case of *Dentify.Me App* it is recommended to have other data outputs, as is usually the case: Mobile and Wi-Fi data. Since the application will be running in a non-controlled environment, no featured protocol could be implemented because Access Points would need to know it.

A direct attack on the Eyewitness is very difficult to avoid, since it cannot be predicted, however, if the witness observes the incident, he or she should act cautiously and calmly, find a safe place, where they can continue observing, but always keeping his safety at bay as that of others.

The Denial of Service Attack is when an attacker or a group of attackers make many requests to one service, with the aim of "saturating" the server and keeping it busy by responding to empty requests, therefore the server is not able to attend any current requests. The countermeasure proposed against this attack is a behaviour monitor, which is usually an Intrusion Detection System (IDS) or an Intrusion Protection System (IPS). This element is represented in the class diagram as an <<BM>> stereotype instance and it is placed in the IncidentReportHandler as it is shown in Fig. 59.

The man in the middle attack (MITM) is where the attacker places himself between two assets which exchange data, the attacker inspects traffic and finds usernames, passwords, or any data sent in plain text. This is the reason why the countermeasure against this attack in the Dentify.Me app study case is to use point to point encryption, in addition to using encrypted network connections (HTTPS or VPN). Within the *Dentify.Me App* study case, it was aimed to encrypt the communications between the mobile device and the server, i. e., between the Eyewitness and the Incident_ReportHandler, this is represented using two extension elements of IoTSecM: <<C>> and <<D>> stereotypes.

For attacks on the database it is firstly considered a SQL injection (SQLi) attack, where an attacker can execute malicious SQL statements that control the database of web applications (Relational Database Management System - RDBMS).

If a SQL vulnerability is exploited an attacker could bypass the web application's authentication and authorisation mechanism in order to observe, modify or delete content from the entire database.

According to OWASP [71] avoid SQL flaws is simple: Stop writing dynamic queries prevents malicious SQL statements from affecting the logic of executed query.

In [71] a set of simple techniques is provided to prevent SQL Injection:

- Use prepared statements
- Use of stored procedures
- Whitelist input validation
- Escaping all user supplied input
- Enforcing least privileges
- Performing whitelist input validation as a secondary defence

For *Dentify.Me App* it is proposed to follow the recommendations given by OWASP although as an addition we propose the database encryption, since the requests to the database are not of an urgent nature, nevertheless the user´s information is very sensitive as well as the lists, therefore the <<C>> instance is proposed to protect the information confidentiality. The <<C>> stereotype will be used as a crypto-module by the Potential_Missing_List, Eyewitness and User Account .

The IoTSecM use case for de *Dentify.Me App* is shown in Fig. 58 where there are six actors: User, Victim, Eyewitness, Rescue Team, DVI Team and Emergency Contact. According to the analysis done before, authorisation is required for the Users, Eyewitness, Rescue Team, Emergency Contact and DVI team, this will protect the system against unauthorised access and as a consequence it will not allow unauthenticated actor to access to the resources. As it was mentioned earlier, a text box over the actor’s head within a Z depicts that the actor must be authorised. The Z element normally

implies that an N instance is implicit. A secure communication ([SC]) requirement is needed as well, in order to indicate that the channel between the user and the account management needs to be secure. SC will protect the information against MITM attacks since the attacker would be able to observe the communication flow, nevertheless they would not be able to understand them. The use case diagram is shown in Fig. 58.

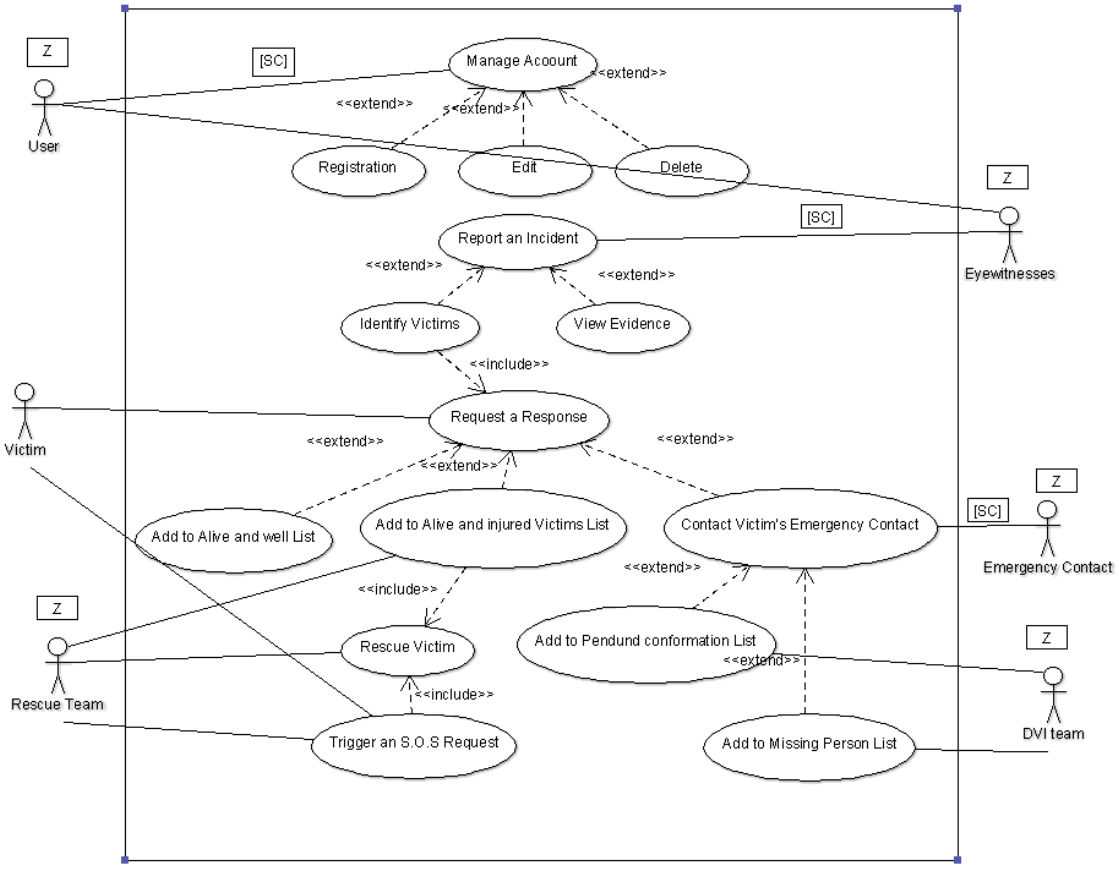

Fig. 58 IoTsecM use case diagram for Dentify.Me App.

The IoTSecM class diagram depicts the system architecture where the security classes are shown, see Fig. 59. Once the threat analysis is done the use case diagram regarding security requirements was obtained, then the IoTSecM class diagram is proposed, where the security requirements are depicted with the functional requirements within the same diagram, that fact makes a better understanding of the security necessities, the costs, and the system reliability. The <<C>> stereotype instance is used by the AM_FORM class to provide a confidentiality requirement;

this same element is utilised by the Potential_Missing_List, Eyewitness and User_Account. The <<Z>> stereotype instance depicts an authorisation mechanism which provides protection against the attacks described earlier. The <<Z>> element authorises the Rescue Team to access the victim information. As it is shown in Fig. 59 the Z element normally requires an authentication mechanism (N element), which helps to first authenticate the actor, once it is authenticated an assertion is passed to the Z element and it will verify its access control list (ACL), or any other control implemented to authorise the actor and guarantee some rights, it is the writing, executing or reading rights.

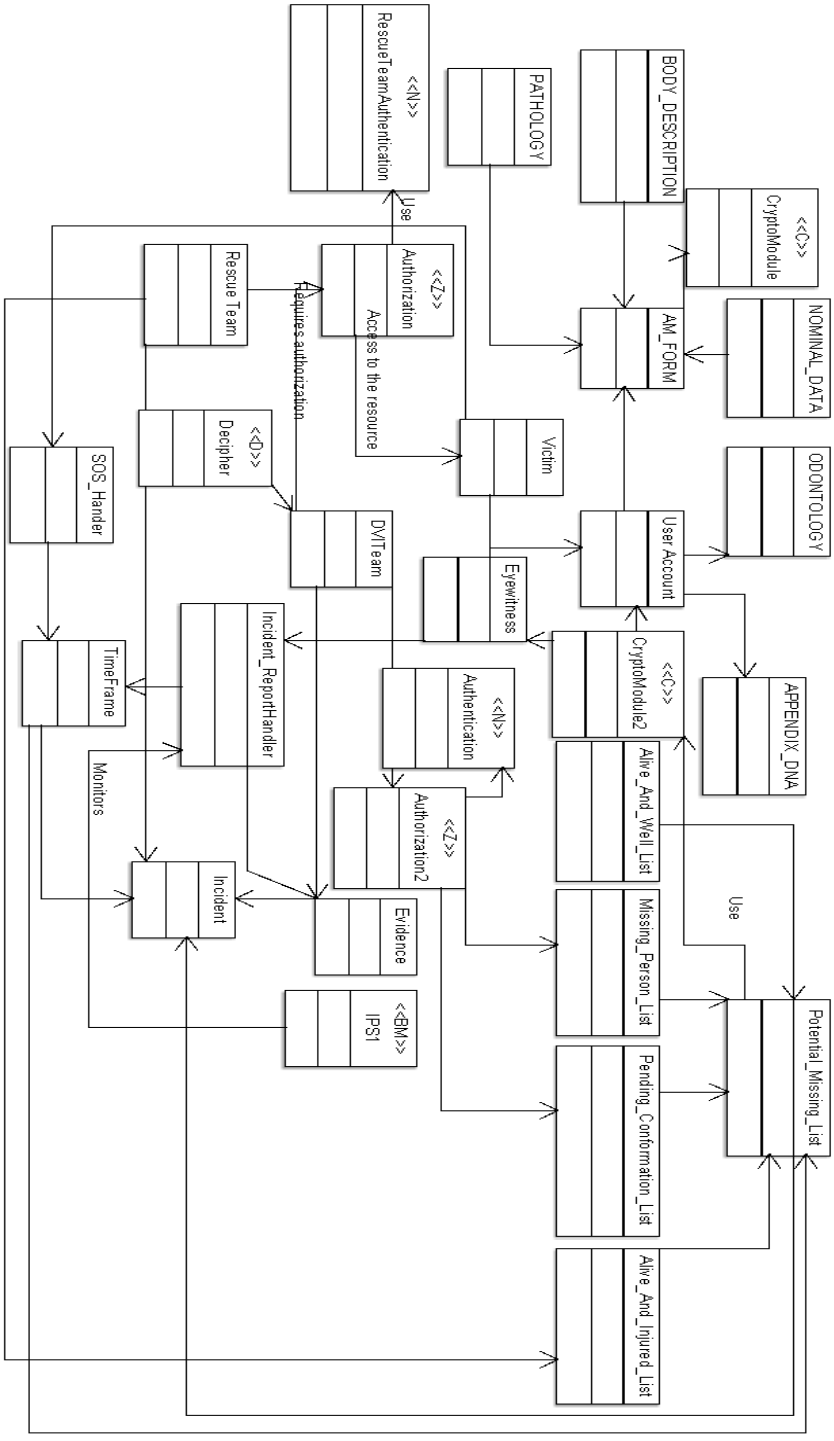

Fig. 59 IoTsecM class diagram for Dentify.Me App.

A <<BM>> stereotype is applied the Incident_ReportHandler, this indicates that an IPS needs to be designed in the next stage of the development life cycle, those security mechanisms will help to prevent the DoS attack mainly.

The IoTsecM approach helped to identify and represent the security and functional requirements in UML/SysML notation. In the *Dentify.Me App* study case a threat modelling was followed which enables the assets identification that are the relevant elements for the system functionality and therefore, for the attackers. Once the assets were identified and the system architecture was obtained, the threat modelling was considered using attack trees. These attack trees modelled the sequence of attacks targeted to a threat, the result of such attacks and sub-attacks are attack vectors which are all the possible ways an attacker can follow to reach the threat. The IoTsecM functionality is to identify the security requirements in order to mitigate the possible attacks. Once the countermeasures are identified and proposed the IoTsecM profile allowed the representation of such countermeasures within the system architecture and the associations.

The security requirements consideration since the analysis stage in both study cases allowed the representation of the proposed countermeasures which will mitigate the attacks regarded in the attack tree. The security analysis must be a fundamental process when IoT systems involve sensitive information, since the consequence of the absence of security mechanisms would imply huge risks as human lives. Therefore, the IoTsecM represents a very useful tool which help to understand and consider the IoT system security before it is implemented in physical objects, the result of such consideration may be more powerful processors in nodes, a refined policies establishment, data encrypted, IDS or IPS implementation, etc. These security controls and mechanisms, definitely, change the system architecture, design and deployment, due to the security mechanisms identified, the companies and developers can save money and lives.

The IoTsecM approach allows the IoT security requirements, and in comparison to the state of the art, it is specific for IoT, thus, it is considers the IoT actors and environment. IoTsecM depicts and models the security requirements and it is a UML/SysML extension, hence it is visual and it helps to the security requirements conceptualization and representation. In comparison with the state of the art there is not any UML/SysML extension which covers all this aspects. This information is summarized in Table XXX.

Table 27 IoTsecM comparison

Extension or Language	Specific for IoT	System security model	UML extension or visual representation
UMLsec [17]	NO	YES	YES
IoT-A [13]	YES	NO	YES
SysML [41]	NO	NO	YES
SysMLsec [33]	NO	YES	YES
UML4IoT [22]	YES	NO	YES
ThingML [74]	YES	NO	NO
FTA-IBM [19]	YES	NO	YES
UML [37]	NO	NO	
IoTsecM	YES	YES	YES

The IoTsecM proposal is well-defined, since it was able to model each one of the countermeasures found in the study cases, the notation comply with the UML extension rules. In both study cases IoTsecM verifies its usability since it modelled and depicted the security requirements, nevertheless, the usability of IoTsecM depends on the designing and modelling team, since IoTsecM is an approach which encapsulates security knowledge within the IoT environment. Therefore, the applicability of IoTsecM certainly will help designers to shrink the attack surface, which is a fundamental fact for IoT systems.

6 CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

The security requirements consideration from the analysis stage must be an essential part for the IoT systems. The IoT systems handle very sensitive information, hence, the implementation of IoT systems regardless the security requirements from the analysis stage should be unthinkable. There was not any tool, modelling language or method to represent the security requirements from the analysis stage, hence the IoTsecM proposal is able to fill that gap.

The IoTsecM proposal aims at depicting the security requirements within the analysis and requirements stages in a MBSE approach, in particular in a waterfall development life cycle. Therefore, the IoTsecM proposal is focused on the representation of the security requirements in a visual and abstract way within a very well-known modelling language which is the UML and the SysML. The IoTsecM approach can represent together the security requirements with the functional requirements.

The security services in an IoT environment are addressed with the IoTsecM nomenclature which is not modelling language-dependant. On the other hand, it is an abstract representation of the security services encapsulated in a nomenclature which incorporates fourteen elements in order to be applied in a visual grammar, or in a modelling language where the notation, constraints and syntax needs to be defined.

The IoTsecM proposal was designed and applied accordingly to the general objective. The IoTsecM UML/SysML profile was designed according to the OMG rules and the UML/SysML exetension mechanisms (stereotypes, constraints and tags). The IoTsecM profile comprises the IoTsecM nomenclature, which was adapted and defined based on the UML/SysML approaches. IoTsecM extends the UML/SysML notation and syntax, this allows a better and easier way to describe security services encapsulated within the nomenclature.

The IoTsecM proposal was tested in two systems, the first one is related to autonomous vehicles and the second one is related to a mHealth App. In the two study cases the IoTsecM proposal used a threat modelling process which follows some steps explained in chapter 5, in that process the IoTsecM proposal helps to represent the security countermeasures obtained, in both study cases, by attack trees. All the countermeasures regarded were depicted applying the IoTsecM profile, hence, it was well-defined and well-formed, and it was applied in UML/SysML tools such as Argo, Papyrus and Ttool. The IoTsecM profile extended the UML notation. In particular the use case, class, component and device diagrams are the most relevant extended diagrams due to the emphasis done to improve the security requirements depicting and conceptualisation in the analysis stage.

The security requirements and countermeasures were depicted together with the functional requirements in the use case diagram and in the system architecture. Therefore, the IoTsecM objective was reached successfully, since IoTsecM proposal was able to depict each countermeasure identified. The hypothesis was corroborated since the IoTsecM proposal helped to the conceptualisation and depicting of security concerns within the analysis stage in a waterfall development life-cycle.

6.2 FUTURE WORK

The IoTsecM profile extends the UML/SysML notation, it depends on the UML/SysML metamodel rules, a future work might be to extend the IoTsecM proposal to a unique metamodel which would be ruled by OMG and not by UML or SysML, it would be a new modelling language dedicated to the security concerns within the IoT systems.

Another future work is to model and simulate the security mechanisms identified in state machine diagrams, this would allow to model the security mechanisms together with the functional requirements in order to see how they should behave and the impact in latency they would have on the system, this is the next step. The IoTsecM proposal is placed in the analysis stage, thus a future work is to extend it to the next stages such as design and implementation. Some tools can be developed to model the behaviour of security mechanisms, where the interaction between the IoTsecM actors, the IoTsecM nomenclature and the functional objects could be seen.

The methodology oriented to security within the IoT systems was a first approach, we identified and proposed some stages, however the methodology needs to be accomplished in order to incorporate the threat modelling, attack diagrams, attacker modelling, etc. Therefore, a complete methodology would help to the integration of IoTsecM with other proposals such as methods, processes, etc., in order to guarantee the constant reviewing of the security concerns in IoT systems. The methodology should consider stages from the security requirements obtaining, until the implementation. Such methodology must consider penetration tests to the deployed system, in order to feed back the system with new security requirements. Once new security requirements appear the IoTSecM proposal can be applied as well, however the procedures and methods to propose the new security requirements is needed, where the peculiarities of IoT systems can be addressed.

There are three well identified activities that can be performed as future work in the IoTsecM context: the modelling language generation, the extension of IoTsecM to the design stage and a whole methodology oriented to the IoT systems security. Those future works would imply the state of the art reviewing about the known methodologies oriented to security, current modelling languages, etc. The understanding and application of OMG and MOF rules for the modelling language proposal. For the IoTsecM extension to cover the design stage, a reviewing of that stage would be needed in order to be able to propose new UML/SysML extensions to the IoTsecM elements already proposed.

6.3 RESEARCH OUTPUTS

Article in MTYmex EAI Conference (D.A Robles-Ramirez, P.J. Escamilla-Ambrosio, R. Acosta-Bermejo, E. Aguirre-Anaya, A. Rodríguez-Mota, J.J Reyes-Torres “Security Oriented Methodology for Designing Internet of Things Systems”, 2017)

- In press.

Research stay

- Application of the UML/SysML extension in a real-life Project (Flourish)

- A paper is in writing process with Dr. Theo Tryfonas and the cryptography lab. In Bristol and for a journal (JCR).

Article in Morelos IEEE - icmeae conference. (D.A. Robles-Ramirez, P.J. Escamilla Ambrosio, T. Tryfonas, “IoTsec: UML extension for Internet of things systems security modelling”, 2017)

Collaboration as principal author in an article proposal for the “Security and Communication Networks” journal (JCR). “Special Issue on Security and Privacy for Smart, Connected, and Mobile IoT Devices and Platforms”. This collaboration is with the MIT doctor.

REFERENCES

[1] IEEE, “Towards a definition of the Internet of Things (IoT),” pp. 1–86.

[2] B. Morin, N. Harrand, and F. Fleurey, “Model-Based Software Engineering to Tame the IoT Jungle,” no. 1, 2017.

[3] R. K. Scott J. Shackelford JD, Anjanette Raymond, Rakshana Balakrishnan, Prakhar Dixit, Julianna Gjonaj, “When Toasters Attack: A Polycentric Approach to Enhancing the Security of Things,” no. October, pp. 1–54, 2009.

[4] M. M. Hossain, M. Fotouhi, and R. Hasan, “Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things,” *2015 IEEE World Congr. Serv.*, pp. 21–28, 2015.

[5] S. Li, L. Da Xu, and S. Zhao, “The internet of things : a survey,” no. April 2014, pp. 243–259, 2015.

[6] A. De Saint-Exupery, *Internet of Things: Challenges and Opportunities*, vol. 291, no. 4. 2014.

[7] B. Russell and D. Van Duren, *Practical Internet of Things Security*. 2016.

[8] “feature-content @ newsroom.cisco.com,” *Cisco.Connections counter. The Internet of everything in motion.* [Online]. Available: https://newsroom.cisco.com/feature-content?articleId=1208342.

[9] Hewlett-Packard, “Internet of Things Research Study 2015 Report,” p. 6, 2015.

[10] A. Salinas, Y. Ben Saied, and D. Level, “Internet of Things Architecture Concepts and Solutions for Privacy and Security in the Resolution Infrastructure,” no. 257521, 2013.

[11] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, “Proposed security model and threat taxonomy for the Internet of Things (IoT) BT - 3rd International Conference on Network Security and Applications, CNSA-2010, July 23, 2010 - July 25, 2010,” vol. 89 CCIS, pp. 420–429, 2010.

[12] “Industrial Internet Reference Architecture,” pp. 1–101, 2015.

[13] A. Serbanati *et al.*, “Internet of Things Architecture, Concept and Solutions for Privacy and Security in the Resolution Infrastructure,” *EU Proj. IoT-A, Proj. Rep. D4. 2*, no. 257521, 2012.

[14] X. Li, Z. Xuan, and L. Wen, “Research on the architecture of trusted security system based on the internet of things,” *Proc. - 4th Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2011*, vol. 2, pp. 1172–1175, 2011.

[15] J. Choi, Y. In, C. Park, S. Seok, H. Seo, and H. Kim, “Secure IoT framework and 2D architecture for End-To-End security,” *J. Supercomput.*, vol. 2, no. i, pp. 1–15, 2016.

[16] G. Sindre and Æ. A. L. Opdahl, “Eliciting security requirements with misuse cases,” pp. 34–44, 2005.

[17] J. Jürjens, “UMLsec: Extending UML for secure systems development,” *Proc. 5th Int. Conf. Unified Model. Lang.*, pp. 412–425, 2002.

[18] J. A. Estefan, “Survey of Model-Based Systems Engineering (MBSE) Methodologies 2 . Differentiating Methodologies from Processes , Methods , and Lifecycle Models,” 2008.

[19] B. P. Douglass, C. Evangelist, and G. T. Ambassador, “How to Develop the Best Architecture for IoT,” 2016.

[20] J. Jurjens, “UMLsec: Extending UML for secure systmes development.”

[21] O. M. Group, “OMG Unified Modeling Language TM (OMG UML), Superstructure v.2.3,” *InformatikSpektrum*, vol. 21, no. May, p. 758, 2010.

[22] 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, 2016.

[23] “Open Mobile Alliance (OMA), Lightweight Machine to Machine Technical Specification, Candidate Version 1.0, 14 Dec 2015. Available on line:” .

[24] “Internet Protocol for Smart Objects (IPSO) Alliance, IPSO Smart Object Committee, IPSO SmartObject Guideline.” .

[25] “CoAP, RFC 7252 Conctrained Application Protocol.” [Online]. Available: http://coap.technology/.

[26] F. Basile, S. Member, P. Chiacchio, S. Member, and D. Gerbasio, “On the Implementation of

Industrial Automation Systems Based on PLC," vol. 10, no. 4, pp. 990–1003, 2013.

[27] M. Unis *et al.*, "Internet of Things – Architecture IoT - A Final architectural reference model for the IoT v3 . 0 Overview of the IoT-A project partners Acronym Full name Alcatel-Lucent Bell Labs France Telefonica Investigacion y Desarrollo SA University of Surrey," no. 257521, 2013.

[28] M. Compton *et al.*, "The SSN ontology of the W3C semantic sensor network incubator group," *J. Web Semant.*, vol. 17, pp. 25–32, 2012.

[29] J. Jürjens, *Secure Systems Development with UML*. Springer, 2004.

[30] U. M. L. White, "Analyze system safety using UML within the IBM Rational Rhapsody environment .," no. June, 2009.

[31] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML : A UML-Based Modeling Language for Model-Driven Security."

[32] R. Matulevi and M. Dumas, "A Comparison of SecureUML and UMLsec for Role-based Access Control."

[33] L. Apvrille and Y. Roudier, "SysML-Sec: A SysML Environment for the Design and Development of Secure Embedded Systems," *Apcosec 2013*, pp. 1–8, 2013.

[34] L. Apvrille and Y. Roudier, "SysML-Sec Attack Graphs : Compact Representations for Complex Attacks."

[35] "Ttool: an open source kit," 2017. [Online]. Available: https://ttool.telecom-paristech.fr/.

[36] T. Secur and B. Tutorial, "The SecurItree," no. August, pp. 1–25, 2006.

[37] O. M. G. Document, N. Normative, A. Normative, M. Consumable, U. M. L. Specification, and S. Rfp, "OMG Unified Modeling Language TM (OMG UML)," vol. 5, no. March, 2015.

[38] "UML 2.5 Diagrams overview," 2017. [Online]. Available: http://www.uml-diagrams.org/uml-25-diagrams.html.

[39] L. Fuentes-Fernández and A. Vallecillo-Moreno, "An Introduction to UML Profiles," *Eur. J. Informatics Prof.*, vol. V, no. 2, pp. 6–13, 2004.

[40] O. Aldawud, "UML PROFILE FOR ASPECT-ORIENTED SOFTWARE DEVELOPMENT," pp. 1–16.

[41] Omg, “OMG Systems Modeling Language (OMG SysML ™) v.1.2,” *Source*, no. June, p. 260, 2010.

[42] S. Friedenthal and R. Steiner, “OMG SysML ™ Specification • Specification status,” 2009.

[43] J. R. C. Nurse, A. Erola, I. Agrafiotis, M. Goldsmith, and S. Creese, “Smart Insiders: Exploring the Threat from Insiders Using the Internet-of-Things,” *Proc. - 2015 Int. Work. Secur. Internet Things, SIoT 2015*, vol. 2015, no. SIoT, pp. 5–14, 2016.

[44] S. Li and T. Tryfonas, “The Internet of Things : a security point of view,” 2016.

[45] B. R. Leidos, C. Garlati, and D. Lingenfeleter, “Security Guidance for Early Adopters of the Internet of Things (IoT),” *Mob. Work. Gr. Peer Rev. Doc.*, no. April, 2015.

[46] “OWASP-Top ten IoT vulnerabilities.” [Online]. Available: https://www.owasp.org/index.php/Top_IoT_Vulnerabilities.

[47] B. Menkus, *Introduction to computer security*, vol. 11, no. 2. 1992.

[48] B. Menkus, “Understanding the use of passwords,” *Comput. Secur.*, vol. 7, no. 2, pp. 132–136, 1988.

[49] B. Russell and D. Van Duren, *Practical Internet of Things Security*. .

[50] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, “Security, privacy and trust in Internet of Things: The road ahead,” *Comput. Networks*, vol. 76, pp. 146–164, 2015.

[51] J. Vollbrecht, P. Calhoun, S. Farrell, G. Gross, and D. Spence, “AAA Authorization Framework, RFC 2904 (Informational),” pp. 1–35, 2000.

[52] D. F. Ferraiolo, R. S. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, “Proposed NIST Standard for Role-Based Access Control,” *ACM Trans. Inf. Syst. Secur. TISSEC Homepage Arch. Vol. 4 Issue 3, August 2001*, vol. 2002, no. 10, p. 338, 2003.

[53] H. K. Das, “Extending UML to include Security Constraints for Embedded Systems.”

[54] S.-W. Lin *et al.*, “Industrial Internet Reference Architecture,” *Ind. Internet Consort.*, pp. 1–101, 2015.

[55] H. Jiang, F. Shen, S. Chen, K. Li, and Y. Jeong, “A secure and scalable storage system for aggregate data in IoT,” *Futur. Gener. Comput. Syst.*, vol. 49, pp. 133–141, 2015.

[56] I. E. Bagci, S. Raza, and T. Chung, “Combined Secure Storage and Communication for the Internet of Things A :,” pp. 523–531, 2013.

[57] I. E. Bagci, M. R. Pourmirza, S. Raza, U. Roedig, and T. Voigt, “Codo : Confidential Data Storage for Wireless Sensor Networks.”

[58] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, “Key management systems for sensor networks in the context of the Internet of Things q,” *Comput. Electr. Eng.*, vol. 37, no. 2, pp. 147–159, 2011.

[59] D. Liu, P. Ning, and R. Li, “Establishing Pairwise Keys in Distributed Sensor Networks,” vol. 8, no. 1, pp. 41–77, 2005.

[60] “Key management,” 2017. [Online]. Available: https://en.wikipedia.org/wiki/Key_management.

[61] M. Bishop, *Introduction to computer security*. Addison-Wesley, 2005.

[62] D. Gambetta, “Can We Trust Trust?,” 2000.

[63] A. Abdul-rahman and S. Hailes, “Supporting Trust in Virtual Communities,” vol. 0, no. c, pp. 1–9, 2000.

[64] F. Bao and I. Chen, “Dynamic Trust Management for Internet of Things Applications,” pp. 1–6, 2012.

[65] M. Hossain, M. Fotouhi, and R. Hasan, “Towards an Analysis of Security Issues , Challenges , and Open Problems in the Internet of Things,” pp. 1–8.

[66] D. Hutchison and J. C. Mitchell, *Lecture Notes in Computer Science*. .

[67] G. E. Suh, D. Clarke, B. Gassend, M. Van Dijk, and S. Devadas, “AEGIS : Architecture for Tamper-Evident and Tamper-Resistant Processing,” pp. 160–171.

[68] S. Raza, L. Wallgren, and T. Voigt, “Ad Hoc Networks SVELTE : Real-time intrusion detection in the Internet of Things,” *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[69] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. De, “A survey of intrusion detection in Internet of Things,” *J. Netw. Comput. Appl.*, vol. 84, no. January, pp. 25–37, 2017.

[70] A. Shostack, *Threat modeling*. John Wiley & Sons, Inc., 2014.

[71] “Implementación simplificada del proceso SDL de Microsoft,” 2010.

[72] “Amenaza,” 2017. [Online]. Available: http://www.amenaza.com/index.php.

[73] S. N. Almutawaa, H. M. Alkabani, M. A. Alsmari, N. H. Alashgar, and A. S. Alrajeh, “Dentify.Me App,” 2017.

[74] N. Harrand and B. Morin, “ThingML : A Language and Code Generation Framework for Heterogeneous Targets,” 2016.