UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DALTON CÉZANE GOMES VALADARES

A TRUSTED EXECUTION ENVIRONMENT-BASED ARCHITECTURE FOR CLOUD/FOG-BASED IOT APPLICATIONS

CAMPINA GRANDE - PB
2020

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Ciência da Computação

A Trusted Execution Environment-based Architecture for Cloud/Fog-based IoT Applications

Dalton Cézane Gomes Valadares

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus Campina Grande como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Metodologia e Técnicas da Computação

Kyller Costa Gorgônio
Angelo Perkusich
(Orientadores)

Campina Grande, Paraíba, Brasil

V136t Valadares, Dalton Cézane Gomes.
A trusted execution environment-based architecture for cloud/fog-based iot applications / Dalton Cézane Gomes Valadares. - Campina Grande, 2021.
123 f. : il.

Tese (Doutorado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2020.
"Orientação: Prof. Dr. Kyller Costa Gorgônio, Prof. Dr. Angelo Perkusich.
Referências.

1. Trusted Execution Environments. 2. Security. 3. Internet of Things. I. Gorgônio, Kyller Costa. II. Perkusich, Angelo. III. Título.

CDU 004.738.5(043)

FICHA CATALOGRÁFICA ELABORADA PELO BIBLIOTECÁRIO GUSTAVO DINIZ DO NASCIMENTO CRB-15/515

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO CIENCIAS DA COMPUTACAO
Rua Aprígio Veloso, 882, - Bairro Universitário, Campina Grande/PB, CEP 58429-900

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

DALTON CÉZANE GOMES VALADARES

A TRUSTED EXECUTION ENVIRONMENT-BASED ARCHITECTURE FOR CLOUD/FOG-BASED IOT APPLICATIONS

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Doutor em Ciência da Computação.

Aprovada em: 27/11/2020

Prof. Dr. KYLLER COSTA GORGÔNIO, UFCG, Orientador

Prof. Dr. ANGELO PERKUSICH, UFCG, Orientador

Prof. Dr. HYGGO OLIVEIRA DE ALMEIDA, UFCG, Examinador Interno

Prof. Dr. DANILO FREIRE DE SOUZA SANTOS, UFCG, Examinador Interno

Prof. Dr. MIRKO BARBOSA PERKUSICH, VIRTUS, Examinador Interno

Prof. Dr. AUGUSTO JOSÉ VENÂNCIO NETO, UFRN, Examinador Externo

Prof. Dr. ALISSON VASCONCELOS DE BRITO, UFPB, Examinador Externo

Prof. Dr. LISANDRO ZAMBENEDETTI GRANVILLE, UFRS, Examinador Externo

---

Documento assinado eletronicamente por **KYLLER COSTA GORGONIO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 08/02/2021, às 17:43, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **ANGELO PERKUSICH**,

PROFESSOR(A) DO MAGISTERIO SUPERIOR, em 09/02/2021, às 05:36, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Lisandro Zambenedetti Granville**, **Usuário Externo**, em 09/02/2021, às 10:09, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **AUGUSTO JOSÉ VENÂNCIO NETO**, **Usuário Externo**, em 09/02/2021, às 10:20, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Alisson Vasconcelos de Brito**, **Usuário Externo**, em 09/02/2021, às 10:25, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **MIRKO BARBOSA PERKUSICH**, **Usuário Externo**, em 09/02/2021, às 10:35, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **DANILO FREIRE DE SOUZA SANTOS**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 09/02/2021, às 10:39, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **HYGGO OLIVEIRA DE ALMEIDA**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 09/02/2021, às 10:48, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **1146385** e o código CRC **DF7A23EE**.

Referência: Processo nº 23096.047175/2020-16 SEI nº 1146385

Resumo

A “Internet das Coisas”, do inglês *Internet of Things* (IoT), é um termo que foi utilizado, pela primeira vez, em 1999, por Kevin Ashton, quando falava sobre a possibilidade de conexão entre dispositivos físicos e a Internet. RFID (Identificação por Rádio-Frequência) foi uma das principais tecnologias usadas naquela época, permitindo identificação e rastreamento de objetos, entre outras aplicações. Desde então, os avanços em várias tecnologias e o surgimento de outras têm possibilitado o barateamento de dispositivos e componentes, despertando ainda mais o interesse da indústria e da academia em explorar as várias possibilidades de aplicações de IoT. Como o uso de tais aplicações vem crescendo, para os mais diversos cenários, torna-se necessário padronizar arquiteturas, protocolos de comunicação e mecanismos de segurança, de modo a facilitar o desenvolvimento de tais soluções e melhorar a confiança dos usuários finais. A falta de padronização ainda é um desafio e, neste sentido, muitas empresas e comunidades de software livre têm proposto *middlewares*, *frameworks* e outras soluções, embora não haja, ainda, nenhum padrão “de facto”, bem definido e aceito. Por causa disto, empresas e pessoas interessadas em tais soluções têm algumas preocupações e dúvidas relacionadas a quais destas devem escolher ou como modelar uma solução específica. Estas preocupações são ainda maiores quando a aplicação lida com dados sensíveis, tais como Informação Identificável Pessoal (PII) ou Informação de Saúde Pessoal (PHI), que demandam proteção e, assim, requerem mecanismos de segurança bem estabelecidos. Nesta tese, o foco é prover uma Arquitetura Confiável para Aplicações IoT. A arquitetura proposta considera autenticação, autorização, criptografia e Ambientes de Execução Confiável (TEE, sigla em inglês). Um Ambiente de Execução Confiável é uma tecnologia fornecida por alguns processadores modernos que habilitam processamento seguro em uma região protegida de memória. A Arquitetura Confiável proposta é validada com um método formal (Rede de Petri Colorida) e com um experimento que mede o desempenho de uma aplicação implementada. Esta aplicação considera alguns componentes FIWARE, para autenticação e autorização, e algumas aplicações Intel Software Guard Extensions (SGX), para processamento protegido. Com a arquitetura proposta, bons níveis de proteção são atingidos, considerando integridade, confidencialidade, privacidade, autenticação, autorização e comunicação segura.

Abstract

The “Internet of Things” (IoT) is a term used, for the first time, in 1999, by Kevin Ashton, when speaking about the possibility of a connection between physical devices and the Internet. RFID (Radio Frequency Identification) was one of the main technologies used in that time, allowing objects tracking and identification, among other applications. Since then, the advances in many technologies, and the arising of many others, have enabled the cost lowering of devices and components, arousing, even more, the industry and academy interest in exploring the many possibilities of IoT applications. As the use of these applications is continuously increasing, it becomes necessary for the most different scenarios to standardize architectures, communication protocols, and security mechanisms to ease the development of such solutions and improve the confidence of final users. The lack of standardization is still a challenge, and, in this sense, many companies and open source communities have proposed middleware, frameworks, and other kinds of solutions. However, there is no “de facto” standard, well defined and accepted, yet. Thus, companies and people interested in using such solutions have some concerns and doubts about which of them to choose or how to model a specific solution. These concerns are even greater when the application deals with sensitive data, such as Personal Identifiable Information (PII) or Personal Health Information (PHI), that demand protection and requires well-established security mechanisms. This work intends to provide a Trusted IoT Architecture (TIoTA) to implement secure IoT applications according to it. The proposed architecture considers authentication, authorization, cryptography, and Trusted Execution Environments (TEEs) to make this possible. A TEE is a technology provided by some modern processors that enable secure processing in a protected memory region. The TIoTA proposed is validated with a formal method (Coloured Petri Net) and an experiment that measures an implemented application’s performance. This application considers some FIWARE components for authentication and authorization, and some Intel Software Guard Extensions (SGX) applications, for protected processing. With the proposed architecture, good protection levels are achieved when considering integrity, confidentiality, privacy, authentication, authorization, and secure communication.

Acknowledgements

To start, I thank God for everything. He blessed everything that I achieved in my life, being with me during every fight and conquers. Then, I thank my mother, an example of honesty, education, comprehension, and strength. She always supported me in achieving my goals during my whole life. I also thank my brothers, my father, my aunt, uncles, and the nearest cousins.

I am grateful to my advisors, professor Kyller Gorgonio and professor Angelo Perkusich, for all the hints, patience, and advice. I keep admiring them both. I thank my previous advisor, professor Andrey Brito, for all the research tips and discussions during the first two years of my doctorate. I also thank all the professors of the examination board for their suggestions and contributions: Alisson Brito, Augusto Venâncio, Danilo Santos, Hyggo Almeida, Lisandro Zambenedetti, and Mirko Perkusich.

My gratitude for the support received from friends and colleagues, as well as the Instituto Federal de Pernambuco, where I work. I thank the members of the Intelligent Software Engineering (ISE) Group, especially Alexandre Costa and Felipe Barbosa, for the great moments during this journey. I am grateful to Alvaro Sobrinho for his support regarding the model checking process. Lastly, I thank the Secure Cloud project and its members, that partially supported me during the second year of this doctorate, especially Matteus Silva.

Contents

List of Abbreviations

AWS - *Amazon Web Service*

CPN - *Coloured Petri Net*

HCPN - *Hierarchical Coloured Petri Net*

HP - *Hewlett-Packard*

IAM - *Identity and Access Management*

IAS - *Intel Attestation Service*

IoT - *Internet of Things*

NIALM - *Non Intrusive Appliance Loading Monitoring*

OS - *Operating System*

OWASP - *Open Web Application Security Project*

PDP - *Policy Decision Point*

PHI - *Protected Health Information*

PII - *Personally Identifiable Information*

PUF - *Physical Unclonable Function*

RFID - *Radio-Frequency Identification*

SGX - *Software Guard Extensions*

TC - *Trusted Computing*

TCG - *Trusted Computing Group*

TEE - *Trusted Execution Environment*

TLS - *Transport Layer Security*

TPM - *Trusted Platform Modules*

List of Figures

List of Tables

Chapter 1

Introduction

The Internet of Things (IoT) concept has attracted the attention of industries, researchers/academy, and different kinds of customers in the last years. This term (IoT) was used for the first time in 1999 [9; 40], by Kevin Ashton, in a presentation in which he described a network connecting physical devices to the Internet. The possibility of interconnecting a high number of devices promotes the creation of various new applications. Some of the IoT advantages are automation, cost, flexibility, and remote management [4]. Thanks to the advances in devices technologies and communication protocols, these advantages enable applying IoT to many domains, such as healthcare, logistics, transportation, and industry.

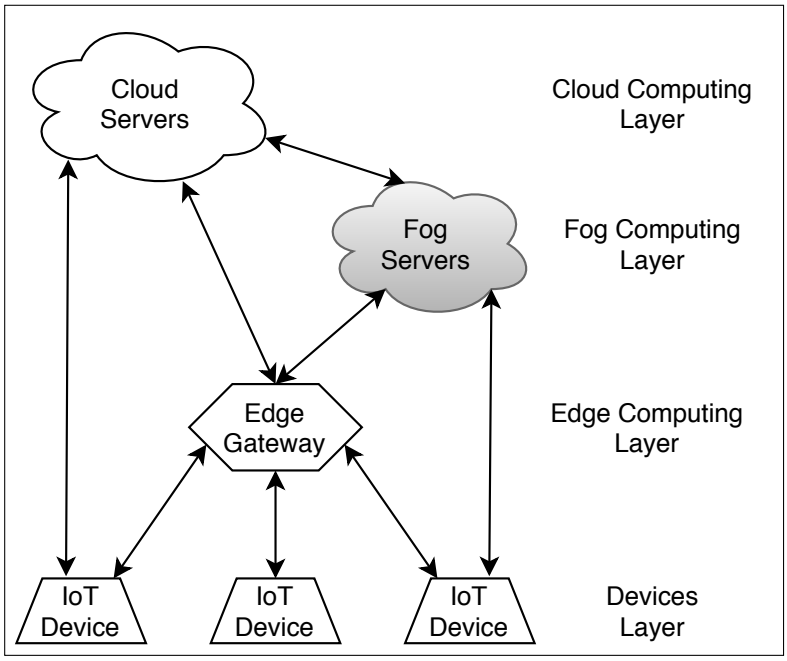
Although the IoT term was born with RFID (Radio-Frequency Identification) projects, which used RFID tags, IoT sensors and actuators are increasingly becoming smarter. Even when they are not so smart (i.e., they have computational resources limitation), they can connect to gateway devices that may perform additional functionality. Due to advances in IoT devices, they are frequently called smart objects [45; 55]. Despite this, the smart objects (IoT devices) often have specific constraints like limited processing capabilities, storage, energy, and connectivity. Thus, many applications require the use of a more robust device, generally called as gateway, with more processing and memory power and fewer constraints.

According to the local where the gateway is deployed and, then, the preprocessing is performed, we have the concepts of edge and fog computing, which are alternatives to cloud computing. When the gateway is located together with the IoT devices, with local data processing, this scenario is named as edge computing [97]. When IoT applications demand even more processing power, with increasing constrained time requirements, instead of sending

data directly to the cloud, specific servers can be deployed locally to provide the required resources for gateways and IoT devices. This scenario is the basis of fog computing, in which IoT devices collect data from the environment and send them to a fog server. Then, the fog server can preprocess these data before sending them to a cloud server or store them locally, for instance [109; 46]. Fog computing can reduce latency when certain situations require fast responses in the field, and it can also decrease costs with cloud processing and communication.

The device's constraints are also dealt with the cloud-based IoT paradigm. The idea, in general, is to send all collected data, previously preprocessed by an edge gateway or a fog server, to be processed and analyzed in the cloud and shared for use in other systems [65]. Among the cloud-based IoT benefits, the following can be highlighted: long-term storage and processing of collected data, data reuse in multiple services, data integration of different users/things, user mobility support, and granted availability for intermittent connection [45].

Figure 1.1: Common Scenarios for IoT Applications



Indeed, the cloud and fog computing paradigms increase the power of the IoT. Such a combination alleviates the concerns regarding the IoT devices typical constraints, such as limited memory and processing capabilities, by allowing applications demanding complex and fast processing to run in fog/cloud servers. Figure 1.1 shows the common IoT scenarios

considering the mentioned paradigms. The main entities are the IoT devices that collect data from the environment. These data can be processed locally in the devices or can be sent to an edge gateway, a fog server, or a cloud server. Whenever using an edge gateway, it can also communicate with a fog or cloud server when the application demands more processing, memory, and storage. The fog servers can also exchange data with the cloud servers. The arrows in Figure 1.1 represent the communication possibilities between the devices and the edge, fog, and cloud layers.

Nowadays, IoT applications can be applied to a large number of scenarios, which goes from healthcare assistance to industrial automation. According to McKinsey&Company, the economic impact of IoT can rise from 2.7 to 6.2 trillion dollars by the year 2025, while Gartner predicted the deployment of 20.8 billion IoT devices by the year 2020 [16] (in 2017, this number was equivalent to 8.4 billion devices in the world, 31% more than in 2016 [110]).

1.1 Motivation

Many times IoT applications have to deal with sensitive data, like Personally Identifiable Information (PII), such as medical, financial, or educational information. Such information can include name, fingerprint, telephone number, social security number, among others, what requires special care regarding the overall data security. A particular type of PII is called Protected Health Information (PHI), which is related to health information from patients in healthcare organizations. This information has become an attractive target for criminals, once the PHI records do not include only health data but also valuable information like address, date of birth, and social security number.

The interest in these data is even higher because they do not use to change, unlike credit card and bank account numbers, which can be canceled. Besides, while credit card numbers can be sold by US$1, on the illicit market, and PII can be sold by US$10 up to US$20, medical information such as PHI can reach between US$20 and US$50, being the most valuable kind of information [60]. The number of PHI breaches reported by the USA federal government in 2016 reached the health information of more than 15 million people, with the biggest case happening in Arizona, through a network server hacking attack, which involved

more than 3.7 million patients' health information [89]. Also, in the same year, a successful ransomware attack forced a hospital in Los Angeles to pay US$17,000 to get its resources and network released [89].

Another known scenario that demands data protection, although it does not deal with PII, is a smart metering application that periodically measures the energy consumption of a given place. If an adversary gets access to a data set containing energy consumption measurements from a specific period (e.g., a month), he/she can use a Non-Intrusive Appliance Loading Monitoring (NIALM) technique to estimate what appliances were used in a specific time and, by doing this, estimates how many people were in that place and what these people were doing [70; 109].

According to Bertino [16], HP (Hewlett-Packard) performed a study with popular IoT devices, showing a high average number of vulnerabilities: 25 vulnerabilities per device, with 80% of devices failing to require good passwords (considering complexity and length), 70% without local and remote encryption for communications, and 60% with vulnerable user interfaces and firmware. For the OWASP (Open Web Application Security Project) IoT project, many IoT vulnerabilities have arisen due to the lack of security techniques adoption, such as encryption, authentication, and access control [16]. According to Gartner, the world-wide spending on security products and services would reach $124 billion in 2019, which is 8.7% more than in 2018 [73].

Frequently, the IoT applications generate or deal with sensitive data related to specific businesses that can involve large financial transactions or even PII/PHI of their users. Both cases, among many others, require special attention to provide security and privacy capabilities for these kinds of applications. It is mandatory to ensure an acceptable level of trust for processing and storing these data, which is an actual challenge nowadays (privacy and security are still concerns related to IoT applications [65; 121; 16]). Furthermore, there are limitations regarding the lack of standardization and limited resources of many IoT devices.

1.2 Problem

Considering that IoT devices can collect information such as location, time, and context, enabling the inferring of personal habits, behavior, and individual preferences, usually there is the need for protection in terms of data acquisition, management, transit, and use/processing. Thus, the sensitive data generated by IoT devices should be protected from unauthorized third parties, resulting in concern to end-users related to loss of control over these data. In general, risks and threats must be considered at the beginning of the applications' development, i.e., during the applications' design, as Privacy by Design[1] [59] and Privacy by Evidence[2] [13] techniques recommend. As it is hard to address all the privacy aspects in the development process, like the Privacy by Design approach proposes, the privacy settings are usually left to the user [45; 119]. The privacy sphere includes the network and all IoT devices that a user owns and trusts to preserve sensitive data [45].

The consequences when an acceptable level of privacy is not reached vary from the non-acceptance of the cloud services to strict costly lawsuits. A current technical challenge is to implement the "right to be forgotten", a data protection regulation proposed by the European Union stating that information about someone has to be automatically removed after some time [45]. Due to the inherently distributed nature of the IoT applications, adversaries use to explore their vulnerabilities for using their devices to generate DDoS (Distributed Denial of Service) attacks [86; 117; 39]. For instance, the number of DDoS attacks increased by 91% in 2017 due to the IoT devices/applications. This fact leads the users to adopt extra effort to ensure security/privacy in IoT applications since often they need to trust the providers. The lack of appropriate security mechanisms in IoT applications is dangerous in sensitive domains, like healthcare, for instance. Unfortunately, there is still no guide for applying a satisfactory level of security in this kind of application [4].

A basic architecture for an IoT application considers a publish/subscribe system respon-

[1] Privacy by Design is an approach that considers privacy concerns during all the development process and demands the adoption of privacy techniques during all the software development cycle.

[2] As the Privacy by Design adoption is considered difficult to enforce in some situations, the Privacy by Evidence methodology proposes a guide to help developers applying privacy techniques according to a well-defined process that contains a checklist, and generates evidence that privacy concerns are considered and mitigated.

sible for registering produced data and distributing accordingly to interested entities. In general, the interested entities subscribe to topics in the publish/subscribe system, enabling them to receive specific data or measurements produced by other entities. Thus, whenever new data are available or changes occur in the measured values already registered, the publish/subscribe system sends notifications to the subscribed entities. After receiving the data, the entities can take important decisions as turning on/off machines, triggering alarms, and performing complex processing.

An instance of this basic architecture can be an energy consumption monitoring application, considering two perspectives to facilitate the understanding: firstly, there are many smart meters, named here *producers*, collecting energy consumption from different buildings throughout a smart city; secondly, there are applications for processing the data aggregation for customers' billing purposes, which are the *consumers*. Some techniques that hit the customers' privacy, like NIALM [35; 70], enable attackers to estimate the number of people in buildings, as well as to gain information on what kinds of activities are performed at a specific time. The same applies to an industrial environment, in which attackers can discover details/secrets related to production, e.g., estimating how many machines are working or how many workers. Because of this, it is highly recommended to keep a satisfactory level of security and privacy in architectures that support such scenarios.

To avoid such type of attack, the producers can encrypt the data ensuring they are processed only in a secure way. A possible approach is the application of homomorphic cryptography [101], which performs common operations over encrypted data. Unfortunately, its high computational overhead makes it unpractical to use with complex operations [83]. Another approach is the use of a Trusted Execution Environment (TEE), that enables, by using specific hardware instructions, the creation of a shielded space in the memory in a way data can be processed securely inside this space. Encrypted data enters the protected area, being decrypted only inside it and encrypted again before leaving. The TEE use is more feasible than homomorphic cryptography because it takes considerably less time to process data [101]. More details about TEE will be found in Chapter 2, Section 2.3.

As seen, there is still a lack of solutions that provide acceptable security levels in IoT applications. The RFC 8576[3] presents an overview of the security requirements for IoT ap-

[3]https://tools.ietf.org/html/rfc8576

plications, discussing threats, challenges and directions based on the state of the art. Privacy protection and data leakage prevention are among the open challenges. Major companies such as Google IoT[4], Azure IoT[5], and AWS IoT[6] only specify the authentication methods required for devices and the communication protocol, i.e., they do not provide other means to protect data. This way, each solution must take care of its security in terms of architecture and development. Regarding this problem, this doctoral work aims at answering the following two research questions:

- RQ1 - How to integrate security components providing a trusted architecture for IoT applications?
- RQ2 - How to limit the need to trust the storage provider and the data consumer in cloud/fog-based IoT applications?

To guide this work, helping to answer these research questions properly, the problem can be defined in two ways:

- Business Problem - define a trusted architecture to develop IoT solutions with an acceptable security level, decreasing the users' concerns with sensitive data;
- Technical Problem - investigate, integrate, and apply security techniques and technologies to protect data in IoT applications, providing secure solutions.

For this work, a solution achieves an "acceptable security level" when it keeps data confidentiality, integrity, and availability, making the data exploitation difficult for an attacker with a high privilege level (e.g., admin level).

1.3 Goals

The main objective of this work is to provide an architecture that developers can follow to improve the data security in IoT applications, mainly in cloud/fog-based IoT, when users commonly have to trust the servers (services providers). The specific objectives are listed below:

[4]https://cloud.google.com/solutions/iot

[5]https://azure.microsoft.com/en-us/free/iot

[6]https://aws.amazon.com/iot/

- apply components for authentication and authorization of the entities;
- apply cryptography to protect data in rest and in transit;
- apply a component for keys management;
- apply Trusted Execution Environments for components that can process sensitive data;
- specify the communication flow to keep data secure;
- create a Coloured Petri Net model with the components and communication flow of the architecture;
- verify security properties through a model checking technique;
- implement a proof of concept application and verify its performance.

1.4 Proposal, Methodology, and Contributions

Considering the concerns regarding data security in IoT applications and data dissemination in fog/cloud computing, we propose a trusted architecture that considers entities' identification, access control, cryptography, and trusted processing. We present a solution that prevents users' data from unauthorized access (e.g., NIALM attacks), keeping them private and secure. To guide us in this direction, we considered the following methodology:

- perform a systematic literature review about the use of TEE in IoT applications (Chapter 3, Section 3.2);
- investigate security components to integrate the architecture (e.g., authentication and authorization components);
- define the trusted architecture considering important security aspects and insights obtained from the literature (Chapter 4);
- evaluate the proposed trusted architecture formally, considering security properties (Chapter 5, Section 5.1);

- implement an application with the proposed architecture and run experiments to evaluate its performance (Chapter 5, Section 5.2).

Based on the literature review (Chapter 3), we decided to adopt trusted execution environments for our architecture once they enable secure processing, even when an attacker has privileged access to the machine. Then, we defined a base case scenario that could need data protection. The chosen base case is a basic architecture commonly employed for IoT applications (Chapter 4, Section 4.1). This architecture has producers, consumers, and a broker to store data. Considering this basic architecture, we investigated how to apply security components to protect data. We defined how to integrate the security components into the architecture and elaborated the communication flow needed to keep data secure. After the trusted architecture definition (Chapter 4), we modeled the trusted architecture using a Coloured Petri Net and verified 14 security properties applying model checking (Chapter 5, Section 5.1). Lastly, we implemented an application as a proof of concept to evaluate its performance (Chapter 5, Section 5.2).

We propose the use of a TEE technology - Intel Software Guard Extensions (SGX) or ARM TrustZone, for instance - for protected processing of sensitive data. To avoid the use of a Publick Key Infrastructure, we also propose a component responsible for keys generation, storage, and management, named Key Vault. The Key Vault also provides data security by using a TEE technology to protect its execution. Both data producers and consumers have access to specific public and private keys from Key Vault. The producers encrypt measurements and send them for a broker component (e.g., publish/subscribe system), and the consumers, which are also TEE applications, receive these measurements through notifications and can decrypt them. Thus, the measurements are protected, in transit and rest, being decrypted and processed only in a secure space inside a TEE application.

For authentication and authorization purposes, respectively, we propose to use an Identity Management (IdM) System and a Policy Enforcement Point (PEP) Proxy. Producers and consumers can be authenticated and receive access tokens. Then, presenting the received tokens in the requests, they can access the architecture's protected components once these tokens are validated. These requests are intercepted by the PEP Proxy, which forwards the tokens for validation in the IdM system. The protected services are the Key Vault and the Broker, which can be a Publish/Subscribe system.

We used Coloured Petri Nets (CPNs) [52; 51] for modeling and verification of the trusted architecture (specifically, Hierarchical CPN). CPNs have been successfully applied for the formal specification, analysis, and verification of different communication protocols, embedded systems, and data networks [87; 10; 49; 93; 103; 116]. We used CPN Tools[7] for editing and construction of the trusted architecture CPN model. We analyzed the model using interactive and automatic simulation as well as message sequence charts automatic generation. Then, we applied model checking, a formal verification technique used to explore states performing verification and validation of behavioral properties [57]. Considering a given model and some properties defined according to the model specification, a model checking algorithm explores all the state space verifying the validity of these properties, if they are satisfied or not. Properties are described in temporal logic, such as CTL (Computational Tree Logic). For the properties checking, we used the ASK-CTL library of CPN Tools, which is a state and action-oriented variant of CTL.

Below, we list the main contributions of this thesis:

- a systematic literature review regarding the use of Trusted Execution Environments for IoT applications;
- a trusted component, Key Vault, that can replace a public key infrastructure;
- a modification to a Policy Enforcement Point Proxy, to improve the provided protection;
- a trusted architecture for IoT applications, minimizing the distrust in service providers that store and process sensitive data;
- a formal validation of the trusted architecture through CPN and model checking;
- a proof of concept implemented according to the trusted architecture;
- an experimental evaluation to analyse the proof of concept performance.

1.5 Document Organization

The structure of this thesis is organized as follows:

[7] http://cpntools.org

- Chapter 2 contains a brief description of the main topics related to this thesis. For instance, we describe the security basic definitions (Section 2.1), and the concept of Policy Enforcement Point Proxy (Section 2.2). Besides, we describe TEEs, as well as the two main TEE commercial technologies available: Intel SGX and ARM TrustZone (Section 2.3). Lastly, we present Coloured Petri Nets (CPN), Hierarchical CPNs (HCPNs), and Model Checking (Sections 2.4 and 2.5);

- Chapter 3 comprises the scientific foundation for this thesis and is divided into two sections. In Section 3.1, we present related works. These works are mainly related to Cloud-based IoT security and privacy challenges and solutions, and related to the application of CPN models and model checking. In Section 3.2, we present the state-of-the-art about the use of Trusted Execution Environments in IoT solutions. To achieve this foundation, we performed a Systematic Literature Review (SLR), which is presented in this Section;

- Chapter 4 comprehends the main ideas of this thesis. Among its Sections, we describe the base case scenario (Section 4.1), the threat model adopted (Section 4.2), the principles used for the trusted architecture proposed (Section 4.3), two trusted components (Sections 4.4 and 4.5), the proposed trusted architecture and its communication flow (Section 4.6);

- Chapter 5 contains the validation of this work. Firstly, we present a formal validation, using an HCPN model for the proposed trusted architecture and performing the verification of security properties by applying a model checking technique (Section 5.1). Lastly, we present an experimental evaluation performed with an application that implements the proposed architecture (Section 5.2). We show the results achieved with the experiments that measured the latency and scalability of the implemented solution;

- Chapter 6 encompasses the conclusion of this document, summarizing the achieved results and suggesting ideas for future works.

Chapter 2

Technical Foundations

This Chapter comprises some useful basic information regarding concepts, technologies and tools to a better understanding of this work. The first Section (2.1) briefly describes some security basic definitions. Then, Section 2.2 presents a brief description about Policy Enforcement Point. Section 2.3 shortly presents Trusted Execution Environments (TEEs) and the two available TEE technologies in the market (Intel SGX and ARM TrustZone). Lastly, Sections 2.4 and 2.5 present the basics of Coloured Petri Nets and Model Checking, respectively.

2.1 Security Basic Definitions

The computer security concerns with protecting sensitive resources in computer systems. This involves applying techniques and methods to preserve the CIA security triad: confidentiality, integrity, and availability. To achieve these properties preservation, the computer systems must consider protecting the sensitive data, in rest or in transit, and also all the resources that can work as an access port for these data. For instance, the systems must regard network and operating system vulnerabilities, which can be explored by adversaries to get unauthorized data access. Below, we list the National Institute of Standards & Technology (NIST) definition of Computer Security [44]:

The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hard-ware, software, firmware, information/data, and telecom-

munications).

The list below briefly presents the classical security goals (confidentiality, integrity, and availability) as well as related concepts relevant to the comprehension of this work.

- Confidentiality: guarantees that unauthorized parties cannot access sensitive information.
  - Privacy: directly related to confidentiality, it guarantees that the information owner has control over its access, allowing or denying its disclosure to specific third parties.
- Integrity: guarantees that unauthorized parties cannot modify sensitive information.
- Availability: guarantees that sensitive information will be available for authorized parties whenever necessary.
- Trustworthiness: guarantees that a party can trust in some available information, from a known source, once it was not accessed or modified by unauthorized parties.
- Symmetric cryptography: in this cryptography mode, the communicating parties use a unique key, commonly named as symmetric key, to encrypt and decrypt sensitive information. The symmetric key must be secure.
- Asymmetric cryptography: in this cryptography mode, the communicating parties use two different keys, commonly named as public and private, to encrypt and decrypt sensitive information. The parties can make the public key publicly available, while the private one must be secure. In general, the algorithms use the public key to encrypt and the private key to decrypt data, although the contrary is possible.
- SSL/TLS: Secure Socket Layer (SSL) is a security protocol that creates an encrypted connection between two communicating parties, protecting the data in transit [98]. For this, SSL employes the use of certificates, which identify the parties and can be checked with Certificate Authorities (CAs). The Transport Layer Security (TLS) is the evolution of SSL, working similarly.
- Authentication: process of identity verification, ensuring that someone or something is indeed the person/thing who claims to be;

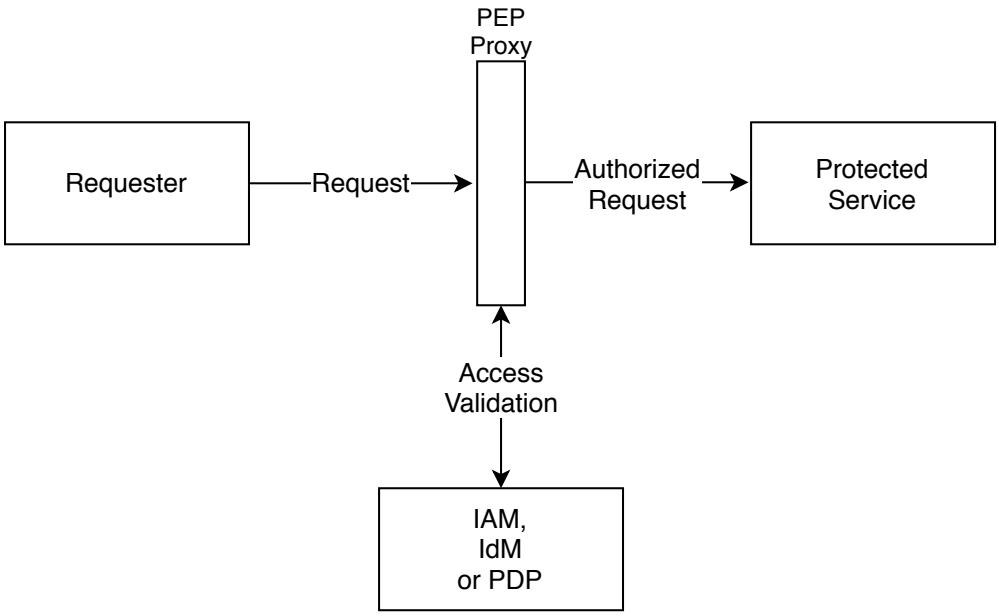
- Authorization: process of denying or allowing access to specific resources (data/system) based on policies.

2.2 Policy Enforcement Point

A Policy Enforcement Point (PEP) works as a gatekeeper controlling the access to a protected service or resource, verifying and validating the requesters to allow or deny their accesses [75]. Generally, a PEP acts together with some policy-based access management system, which can be a Policy Decision Point (PDP) or an Identity and Access Management (IAM) system (also called Identity Management - IdM).

Figure 2.1 shows a common architecture using a PEP. When the PEP proxy receives a request, it checks the access policies with the PDP, IdM or IAM system whether the requester has permission to access the protected resources. Since the PEP enforces the access policies application, it forwards the request to the protected service or sends an unauthorized message to the requester, respectively, whether the access was allowed or denied.

Figure 2.1: Architecture with a Policy Enforcement Point



2.3 Trusted Execution Environment (TEE)

The TEE concept arose with the need to protect data processing in the increasingly complex systems. This technology is a new approach for Trusted Computing (TC), which was developed by the Trusted Computing Group (TCG), in the earlier 2000s, to provide better levels of secure computation, privacy and data protection. The TC was initially implemented by using Trusted Platform Modules (TPM), which are separate tamper-evident hardware modules for platform security that allow cryptographic keys protection and data integrity. Since TPMs do not allow third-parties run code inside them, TEE was proposed to provide an isolated protected execution environment for third-parties applications.

TEE can be defined as a tamper-resistant processing environment, running on a separated kernel, providing a good level of authenticity, integrity and confidentiality for the executed code [90]. A TEE should supply a remote attestation process, enabling third-parties to prove its trustworthiness. Unfortunately, TEE is not a bullet-proof solution for systems security, since some refined attacks (e.g. Side-Channel[1]) can be used for unauthorized data obtainment attempts. Since 2010, Global Platform[2] is responsible for TEE standardization through its TEE System Architecture and API specifications, which comprise TEE Client API, TEE Internal Core API, TEE Secure Element API, among others[3].

2.3.1 Intel Software Guard Extensions (SGX)

Intel SGX is a hardware-assisted Trusted Execution Environment (TEE) technology that protects code and data from disclosure or modification [47]. Applications intended to be safe executes inside protected memory regions such that their code and data are isolated from other software running in the system, even with higher privilege, like Operating System (OS). These special regions in the memory are called *enclaves*, which are created and manipulated through a distinct set of processor instructions, with the help of a software development kit (SDK) provided by Intel. With these hardware-based capabilities, Intel promises that code

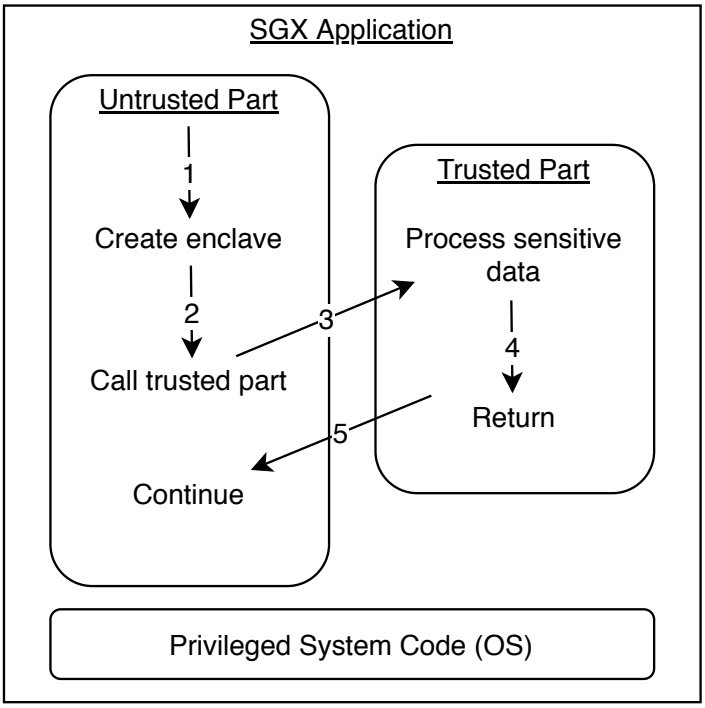
[1] Attacks based on characteristics of the hardware implementation, such as timing information, power consumption, electromagnetic leaks and sound, which require expert technical knowledge about the internal operation of the system.

[2] `http://globalplatform.org/`

[3] `https://globalplatform.org/specs-library/?filter-committee=tee`

and data remain protected even when drivers, OSs or BIOS are compromised.

Figure 2.2: SGX application execution [48]



The execution of an SGX application is shown in Figure 2.2, according to the following flow:

1. the SGX instantiates the application, creating an enclave;
2. the untrusted portion of the code calls the trusted part, which runs in a protected region of memory;
3. the application performs any protected processing inside the trusted part of the application;
4. the trusted part returns the result of the trusted processing to the untrusted part of the application;
5. the untrusted part of the application proceeds the execution.

SGX works with a very small attack surface, formed basically by the processor boundaries, preventing direct attacks on executing code or sensitive data in memory. The enclaves

work in these boundaries, shielding the data and code inside them, and encrypting data when they need to leave the enclave. When data return to enclaves, integrity checking is performed.

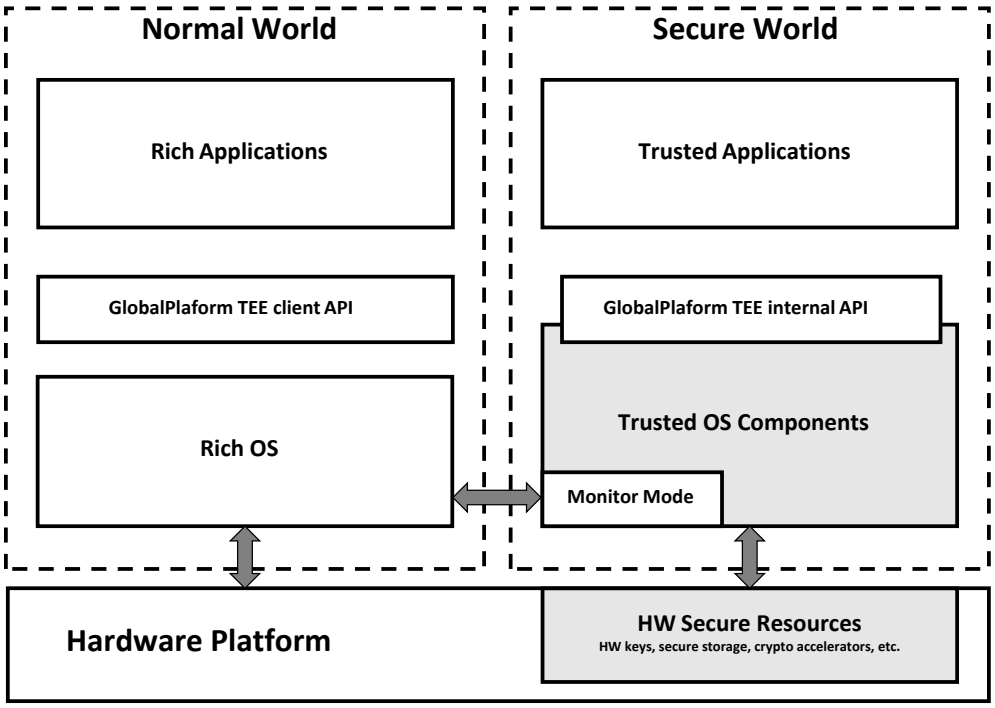
Intel SGX also provides a way to enable remote parties to check if an application executes in a valid enclave of a real Intel SGX processor. This mechanism ensures the authenticity of an enclave, validating, for the remote party, the application enclave's identity. This is possible due to a remote attestation process, which follows a specific protocol: both parts exchange information in a way the remote application can verify the authenticity of the supposed SGX application by accessing the Intel Attestation Service (IAS) and checking the received information, generated in the enclave. At the end of the remote attestation, both sides - the remote application and the application running in the attested enclave - will have a symmetric shared key that they can use to exchange sensitive information with each other. The code authenticity can also be checked, but the description of this process is out of scope for this work. In Chapter 3, Section 3.1, we present some works that use Intel SGX in theirs solutions.

2.3.2 ARM TrustZone

TrustZone is the Arm TEE technology, which provides system-wide hardware isolation for trusted applications [8]. It is more suitable for IoT devices, due to its processor design, based on the ARM architectures, which are among the most used for embedded systems and devices in general, such as micro-controllers, mobile phones, and tablets. The TrustZone applications are divided into two worlds, one secure and one insecure. The secure world runs a trusted OS, responsible for isolating and running trusted applications, providing confidentiality and integrity to the system. The insecure world runs an untrusted OS, normally called Rich OS, which is a common OS such as any Linux distribution. Figure 2.3 shows the TrustZone basic application architecture.

An untrusted software can not access data and resources from the secure world. As seen in Figure 2.3, both untrusted and trusted applications use, respectively, the GlobalPlatform TEE client API and TEE Internal Core API specifications. TrustZone is used in billions of devices applications to protect code and sensitive data in processes such as authentication, payment, and content protection [8]. Some works that use TrustZone are mentioned in Chapter 3, Section 3.1.

Figure 2.3: TrustZone Basic Application Architecture [8]



2.4 Coloured Petri Nets (CPNs)

A CPN is a 9-tuple $CPN = (\sum, P, T, A, N, C, G, E, I)$ [51; 52], where:

- $\sum$ represents a finite set of **colours**, which are non-empty types;
- P is a finite set of **places**, which are representations for the states;
- T is a finite set of **transitions**, which are representations for the events between the states;
- A is a finite set of **arcs**, satisfying $P \cap T = P \cap A = T \cap A = \emptyset$ and indicating the communication flow between places and transitions;
- N is a **node** function, defined from A into $P \times T \cup T \times P$
- C represents a **colour** function, defined from P into $\sum$;
- G is a **guard** function, defined from T into expressions such as $[\forall t \in T : Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \sum]$;

- E is an **arc expression** function, defined from A into expressions such as $[\forall a \in A : Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \sum]$, with $p(a)$ being the place of $N(a)$;

- I represents an **initialization** function, defined from P into closed expressions such as $[\forall p \in P : Type(I(p)) = C(p)_{MS}]$.

Hierarchical Coloured Petri Nets (HCPNs) [52; 51] allow the modeler to structure the model, by modularizing it and introducing interaction mechanisms to build large structured models. An HCPN model has manageable parts, named modules, with well-defined interfaces. HCPNs are useful to abstract some parts of a large system, enabling a more focused analysis, without unnecessary details, due to the different abstraction levels. The HCPN modules have a specific type of transition, named *substitution transition*, that substitutes a detailed module, and a specific type of place, called *port place*, that acts as a buffer to send or receive tokens among the different modules, through the *substitution transitions*.

A CPN module is a 4-tuple CPN_{mod} = $(CPN, T_{sub}, P_{port}, PT)$ [6], where:

- $CPN = (\sum, P, T, A, N, C, G, E, I)$ is a CPN model (non-hierarchical);

- $T_{sub} \subseteq T$ represents a set of substitution transitions;

- $P_{port} \subseteq P$ represents a set of port places;

- $PT : P_{port} \rightarrow In, Out, In/Out$ represents a port type function, responsible for assigning a port type to each port place.

An HCPN model is a 4-tuple $HCPN = (S, SM, PS, FS)$, where:

- S is a finite set of CPN modules, with a requirement that, for each $CPN_{mod} = (CPN, T^s_{sub}, P^s_{port}, PT^s)$, $(P^{s_1} \cup T^{s_1}) \cap (P^{s_2} \cup T^{s_2}) = \emptyset, \forall s_1, s_2 \in S : s_1 \neq s_2$;

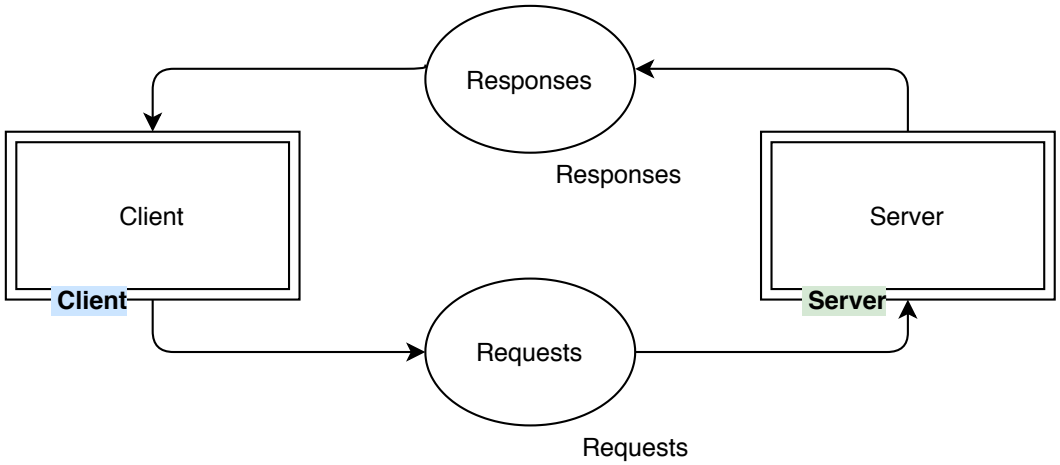
- $SM : T_{sub} \rightarrow S$ is a submodule function, assigning a sub module to each substitution transition, with a requirement that the module hierarchy be acyclic;

- PS is a port-socket function, assigning a port-socket relation $PS(t) \subseteq P_{sock}(t) \times P^{SM(t)}_{port}$ to each substitution transition t, with a requirement that $ST(p) = PT(p'), C(p) = C(p')$, and $I(p)() = I(p')(), \forall (p, p') \in PS(t) \wedge \forall t \in T_{sub}$;

- $FS \subseteq 2^p$ is a set of non-empty fusion sets with $C(p) = C(p')$, and $I(p)() = I(p')(), \forall(p, p') \in fs \wedge \forall fs \in FS$.

For the above definitions, consider the top-level CPN module shown in Figure 2.4, which consists of two substitution transitions (drawn as rectangles with double-lined borders) representing the entities defined for Client and Server. Both substitution transitions replace the Client and Server detailed models. The detailed Client model interfaces with the Responses and Requests places through two port places: an input (IN) port place to receive tokens from the Responses place and an output (OUT) port place to send tokens to the Requests place. Likewise, the detailed Server module interfaces with the two places through two port places (an input port place to receive tokens from the Requests place and an output port place to send tokens to the Responses place).

Figure 2.4: HCPN example



2.5 Model Checking

Model checking is an automatic computer-assisted technique for the analysis and verification of dynamic systems modeled by finite state-transition systems [27]. This process involves the definition of properties based on the model specification. In general, these properties are defined according to model behaviors that must or must not occur along time, which represent partial states of the finite state space. To specify the properties, a logical formalism should be used, such as temporal logic, which can be used to describe how the model

behavior evolves.

The systems are represented as state-machine models, and their executions check the satisfiability of these properties. To verify if a property is valid or not, indicating that a desired behavior is achieved or an undesired behavior is not achieved, a model checker performs an exhaustive search in the state space of the modeled system. The checking procedure should always finish with an answer "yes" or "no".

In short, the model checking methodology comprises the following steps:

1. The system to be analyzed and verified is modeled into a state-transition graph K, usually a Kripke structure K;

2. A property related to the system specification is expressed as a temporal-logic formula φ;

3. The model checker decides whether K is a model of the formula φ, $K \models \varphi$.

A Kripke structure is a finite directed graph in which vertices are called "states", and edges are called "transitions". These structures represent the possible configurations or configuration changes in a discrete dynamical system. In case the structure K is not a model of the formula φ, $K \not\models \varphi$, the model checker should present a counterexample that represents the violation of φ by K. This process indicates that a falsification (e.g., the detection of a problem) is often identified faster than a verification (e.g., the proof of the problem absence).

To represent the properties with temporal logic formulas, linear and branching-time logics are frequently used, such as Linear Temporal Logic (LTL) and Computational Tree Logic (CTL), since both combine linear-time operators and path quantifiers [18]. In linear-time logic, the analysis is performed over the execution paths individually, while in branching-time logic the computation is viewed as a tree and the analysis is performed over the computation tree. This is suitable to specify concurrent systems, once they can represent the order of events along time without explicitly introducing time.

CTL* is a very expressive propositional modal logic that combines both branching and linear-time operators, being more powerful than CTL and LTL [26; 27]. CTL* uses path quantifiers and temporal operators which are interpreted, respectively, over states and paths, and are listed below:

- Path quantifiers
  - **A** - for all computation paths from this state;
  - **E** - for some computation path from this state (there is a path);
- Temporal operators
  - **X** ("next time") - asserts that a property holds in the next state of the path;
  - **F** ("eventually" or "in the future") - asserts that a property holds at some state of the path in the future;
  - **G** ("always" or "globally") - asserts that a property holds at all states of the path in the future;
  - **U** ("until") - combines two properties, asserting that the second property holds at some state of the path in the future and the first property holds for every preceding state of the path, i.e. the first property holds until the second property holds.

Recursively, considering a set of atomic propositions A and an atomic proposition p, the syntax of CTL* can be defined as follows [27]:

- If $p \in A$, then p is a formula of CTL*;
- If φ and ψ are formulas of CTL*, then $\varphi \wedge \psi$, $\varphi \vee \psi$, $\neg\varphi$, $\mathbf{A}\varphi$, $E\varphi$, $X\varphi$, $F\varphi$, $G\psi$ and $\psi U \varphi$ are formulas of CTL*.

For the checking of properties, we used ASK-CTL, a state and action-oriented variant of CTL supported by the CPN Tools. ASK-CTL is an extension of the CTL temporal logic, having a model checker that considers the states and transitions information. This model checker allows the checking of temporal logic formulas considering the model state space, verifying if the defined properties are satisfied. As both ASK-CTL model checker and logic are implemented in SML (Standard ML), the queries related to properties verification are described in SML [120; 25].

Chapter 3

Scientific Foundations

The theoretical foundation for this thesis is presented in this Chapter. It is composed of related works, which are presented in Section 3.1, and the state of the art, presented in Section 3.2. The Related Works Section contains more general references, organized into four Subsections according to the following topics: data security and privacy in cloud-based IoT applications (Subsection 3.1.1); technical approaches to improve security in IoT applications (Subsection 3.1.2); general solutions using TEEs (Subsection 3.1.3); and verification with Coloured Petri Net models and model checking (Subsection 3.1.4). The State of the Art Section contains references more specific to the use of TEEs to protect data in IoT applications. To establish the state of the art, we carried out a Systematic Literature Review (SLR). We describe all the steps performed for the SLR, in Section 3.2, including the applied protocol (research strategy, search string, papers selection methodology, papers quality assessment and data extraction methodology) and the results.

3.1 Related Works

This Section contains the main insights of some works related to security and privacy in IoT applications in general, exploring the existent challenges in this area, and presenting approaches to protect data when IoT applications need to deal with sensitive data.

3.1.1 Security and Privacy Concerns in Cloud-based IoT

Weber [118] presented the concerns related to data privacy in the Internet of Things, mentioning the main challenges and efforts performed by the EU as an attempt to regulate privacy and data protection. This work also treated general security and privacy requirements, privacy-enhancing technologies, and privacy by design methodology, presenting the main aspects that should be taken into consideration. He concluded that the main concern is related to the management of the great quantity of generated data in order to secure storage and communication.

Bertino presented [16] concepts and motivations regarding security and privacy on IoT applications, involving big data and cloud. She highlighted the importance of updates in security and privacy techniques, as well as the creation of new ones since IoT devices have constrained resources that can expose vulnerabilities, which can be explored by adversaries. A problem to deal with big data is the processing of large volumes of unstructured and structured data in continuous flows from a large number of sources. The author also mentioned challenges and directions related to data confidentiality and privacy on big data.

Werner et al. [119] presented a survey with works related to cloud identity management and strategies on how to achieve good levels of privacy, listing main features and challenges. The success of cloud identity management systems is based on trust, access policies, and agreement on security mechanisms with user participation. Due to the intense exchange of sensitive data between service providers, there are always concerns about the privacy of these data. Privacy can be defined as the feature of people having control over their data, in terms of their desired protection. Some of the listed challenges in cloud identity management are identity provisioning/de-provisioning, authentication in federated environments, data privacy, and authorization/attributes management. The authors made a systematic study related to privacy and identity management in clouds and proposed that user Personally Identifiable Information (PII) should be encrypted to prevent non-authorized users from using it. Their proposal is user-centered, giving PII control to users that own the data, assisting users in data dissemination, and helping them about their preferences guarantees on the service provider side.

Alsubaei et al. [4] proposed a framework to assess IoT service providers, regarding

security and privacy, using more than 250 assessment attributes. The considered environment has four main components: edge, gateway, back end, and mobile. The assessment criteria are based on the goal-question-metric (GQM) approach [111]. The framework is divided into three stages: the definition of an IoT SecLA (Security Level Agreement), the security and privacy quantification, and the ranking. To validate the framework, the authors presented a case study, evaluating security and privacy metrics for three service providers and according to three users. The framework was considered simple and efficient, being responsible for quantifying the security of IoT service providers based on customer's requirements.

Henze et al. [45] presented an approach that allows users to set their privacy requirements before they send sensitive data to the cloud, with an adaptable interface in a transparent way. A developer in the development process addresses this privacy functionality. The UPECSI (User-driven Privacy Enforcement for Cloud-based Services in the IoT) solution provides privacy guarantees focused on individual end-users and developers of cloud services. Two application areas were considered: assisted living, in which a lady has control over many sensors collecting data to her building automation system, and public mobility assistance, in a way people with limited mobility can travel independently with the help of assistance systems and public transportation. This approach is user-centered, with each user owning and operating one or more smart objects (IoT devices). It provides data security with adaptable user-control, considering different knowledge of the users, and transparency by design with privacy being integrated into the development process. The solution has three parts: model-driven privacy, interaction with the user, and privacy enforcement points. The Privacy Development Language was introduced to ease the description of privacy considerations. Commonly, users have to accept policies and use the services or do not accept and do not use them. UPECSI offers more flexible decisions, allowing the user to use only parts of a service, according to different customized privacy policies.

3.1.2 Different Approaches to Improve Security in IoT Applications

Aman et al. [5] also presented IoT security challenges, but considering the use of Physical Unclonable Functions (PUFs) to deal with them. As many IoT devices have no enough processing power for many of the security schemes and it is not recommended, for many cases, managing and storing secrets in these devices, PUFs arise to assist in the IoT security. PUFs

enable the identification of integrated circuits using a challenge-response function that receives, as input, the devices' physical micro-structures. Due to the variability in the physical factors during devices fabrication, it is very unlikely the micro-structure replication, which makes these devices unique in this sense. A PUF can be used to protect devices without the need to store secrets. The authors proposed a protocol for mutual authentication to demonstrate PUF working and its advantages in an IoT scenario. Among the advantages, PUF can be used in authentication processes and can improve trust in the devices. The PUF based mutual authentication protocol is presented, making considerations regarding the network model and its assumptions. Furthermore, two analyses are made related to the security and performance of the proposed solution, concluding that PUFs can be used to develop effective and efficient security protocols for IoT applications.

A secure communication channel for IoT devices was proposed by Li et al. [65], based on a heterogeneous ring *signcryption* scheme. This heterogeneous environment involves sensors using identity-based cryptography to send messages for servers in a public key infrastructure. The solution provides confidentiality, integrity, authentication, non-repudiation and anonymity. The authors measured its performance, comparing it with four other techniques with respect to computational time and energy consumption. The proposed solution proved more efficient than others, being suitable for data transmission in IoT.

Wilson et al. [121] attempted to the fact that security solutions are using TLS connections for protecting the communication of sensitive data between devices and cloud services. However, it is difficult for users to know what real data are being transmitted since they mistrust the vendors. The TLS-RaR (Rotate and Release) was proposed, with no modifications required to TLS protocol, allowing users to audit the messages that were exchanged between the devices and the server. This was possible by doing a key rotation in the key used previously for data encryption and a key release for the next communications. This system had the following requirements: past auditability, present-moment integrity, audit robustness and TLS compatibility.

As a consequence of these four requirements, the TLS-RaR cannot audit traffic in real-time. To be evaluated, the TLS-RaR was implemented in an IoT device and a web service. The results presented a CPU overhead of roughly 22%, which can be decreased in IoT devices with hardware support for hash algorithms.

3.1.3 General Solutions using Trusted Execution Environments

3.1.3.1 Intel SGX solutions

In Pires et al. [83], Intel SGX was used to implement a Secure Content-Based Routing (SCBR) system, providing privacy-preserving to messages, since they are not exposed to unauthorized parties, and they are filtered only in a secure enclave. Extensive experiments concluded that SGX adds a limited overhead providing much better performance when compared to other alternatives of Secure CBR. The only problem happens when large workloads are processed into the enclave because the page faults cause high degradation (this is an Intel SGX limitation).

Milutinovic et al. [71] presented a blockchain that uses a proof of lucky consensus protocol. The solution uses a random number generation based on a Trusted Execution Environment, achieving low-latency in the transaction validation, low energy consumption, and deterministic confirmation time. The authors applied the Intel SGX capabilities for this and explained the protection provided by this solution.

Aublin et al. [11] presented TaLoS, a library that reduces the amount of Intel SGX enclave transitions and securely finishes TLS connections inside an enclave, maintaining secure code and data. Security and privacy are addressed with ease-of-deployment. Through experiments, the authors concluded that the reduction in the enclave transitions and the transition overhead resulted in a low-performance overhead with the bottleneck being the TLS handshakes.

Liang et al. [67] proposed the use of Intel SGX and blockchain to protect sensitive health data, achieving accountability for data access. A Personal Health Data Management system is proposed, with a user-centric approach, allowing patients to collect and manage their health data. According to the authors, the proposal achieves self-sovereign data ownership, permanent data record with integrity, scalable processing, decentralized and distributed privacy, and access control, and trusted accountability.

Sampaio et al. proposed a data dissemination platform. [91], providing data security and privacy levels. The proposed solution gives full control to data producers over consumers' access; i.e., producers allow or deny the consumers access to sensitive data and set the granularity level. Thus, sensitive data can be produced and repeatedly anonymized or aggregated

by trusted entities, using Intel SGX, and then consumed by untrusted applications, ensuring the privacy of original data. The use of Intel SGX makes the solution more feasible than homomorphic encryption. The authors evaluated the solution, which achieved a lower overhead and can be useful for medium-scale systems, with small volumes of data dissemination at time.

Nguyen et al. [74] proposed LogSafe, a distributed, scalable, fault-tolerant, and trusted logger for IoT devices data. Using Intel SGX, the proposed logger satisfies confidentiality, integrity, and availability, and also provides tamper detection, being able to protect against replay, injection and eavesdropping attacks. Experiments demonstrated that LogSafe has high scalability, which allows it to work with a high number of IoT devices and a high data transmission rate.

Ayoade et al. [12] proposed a decentralized system for data management in IoT applications using blockchain and TEE technologies. The idea is enforcing access control by using blockchain smart contracts and storing only data hashes in the blockchain, while the raw data are stored in a TEE application. The authors implemented the proposal using Ethereum blockchain and Intel SGX and performed experiments regarding the processing costs at blockchain and SGX application (gas usage, throughput, and CPU time considering the main operations). The obtained results demonstrate that the solution has an acceptable efficiency.

Peters et al. [79] proposed BASTION-SGX, architectural support for Bluetooth trusted I/O using Intel SGX. This work has the goal of protecting I/O data even when considering adversaries with high-level privileges. Authors described challenges regarding the design and implementation of "trusted I/O", presenting a possible solution and describing the implementation of a proof-of-concept, which extends the existing Bluetooth security to an SGX enclave, securing the data between it and the Bluetooth controller. This was done through a secure tunnel between an SGX enclave and the Bluetooth hardware.

Silva et al. [100] presented an architecture for data aggregation in cloud computing, considering two approaches for data security and privacy. The proposed architecture comprises four main components: message bus, producers, aggregators, and consumers. To validate the proposal, the authors implemented proofs of concept and evaluated them regarding the response time to process routine operations in smart metering, such as instant energy con-

sumption and monthly bill calculations. Two different aggregators were implemented: one considering the Intel SGX technology and one considering a homomorphic encryption technique. Tests were performed considering the host machine, virtual machines, and containers. The achieved results demonstrate that Intel SGX enables lower response times than the homomorphic encryption technique. The authors also presented advantages and disadvantages for each approach and presented a security analysis for both.

3.1.3.2 ARM TrustZone solutions

Yang et al. [128] presented Trust-E, a trusted embedded operating system architecture, compliant with Global Platform TEE specifications. The authors designed and implemented the Trust-E solution and implemented a mobile payment application as a demo to test their solution regarding correctness and effectiveness. According to the authors, the results demonstrated that the proposed solution could effectively meet the security requirements.

Lesjak et al. [64] proposed a security solution for industrial maintenance scenarios, designing and implementing a device snapshot authentication system. This solution was implemented with ARM TrustZone and Security Controller, comparing both technologies. The results indicated that the TrustZone solution presents greater flexibility and performance, while the Security Controller solution presents better protection against physical attacks. Authors concluded that the chosen technology depends on the use case and proposed a hybrid approach, using both technologies, which maximizes performance and security: software components that demand more processing power are implemented with TrustZone, while software components that demand more security are implemented with the Security Controller.

Zhang et al. [130] presented a systematic study about a cache incoherence behavior between normal and secure worlds, in the ARM TrustZone, and proposed a rootkit called Cachekit to show the feasibility of including malicious code in the processor cache, keeping it hidden. This was possible due to the incoherent state between normal and secure worlds, allowing the rootkit to evade introspection from detection tools. The authors compared the Cachekit with other rootkits regarding detection methods, and it proved to be the best, without any detection, since the malicious code is completely hidden inside the cache.

Dai and Chen [33] proposed the design and implementation of OPTZ (Open TrustZone),

a multitask hardware isolation between normal and secure worlds, with an architecture composed of a secure OS (for the secure world), a common OS (for the normal world), secure services and a communication mechanism. The authors focused on the communication between normal and secure worlds through the secure monitor, considering a system interrupt design and a multitask hardware isolation. They implemented the architecture with the TrustZone technology and carried out experiments to verify its correctness, testing the physical memory access. The results demonstrated the effectiveness of the proposed hardware isolation.

Park and Kim [77] proposed a trustworthy management system for TCB (Trusted Computing Base) measurements from IoT applications. Authors called the solution TM-Coin, which uses TrustZone and blockchain. They presented the protocol and transactions flow to securely distribute the TCB measurements in the blockchain. TrustZone was used to generate and protect the TM-Coin transactions containing the TCB measurements. The solution applied a remote attestation mechanism and had its performance overhead evaluated through experiments carried out with an implemented prototype.

Pinto et al. [82] proposed LTZVisor, a hypervisor that uses TrustZone to assist virtualization. The authors presented the hypervisor architecture and details of its implementation, which was experimentally evaluated considering three metrics: memory footprint, performance overhead, and interrupt latency. The experimental results demonstrated a low-performance overhead, satisfying strict requirements for real-time environment virtualization when running unmodified rich operating systems.

Chang et al. [19] proposed the use of TEE to address runtime security problems efficiently since it uses hardware isolation technology. They presented the TrustZone architecture, explaining its basic working, which is divided into normal and secure worlds (untrusted and trusted, respectively). They tested a TrustZone-enabled hardware device, evaluating the proposal, which achieved experimental results demonstrating its effectiveness and feasibility.

Yalew et al. [34] presented TruApp, which validates the authenticity and integrity of a mobile app by checking measurements and static/dynamic watermarks, and a verification key issued by the TruApp provider or the app vendor. TruApp is protected since it executes mostly in a TrustZone secure world and verifies the integrity of the part running in the insecure world. The authors implemented the proposal and carried out experiments, concluding

that the measurements have more ability to detect apps that are not authentic than the watermarks, but they also have a higher overhead. They proposed to analyze means of optimizing the TruApp as future works.

Guan et al. [41] proposed TrustShadow, a new system to shield legacy applications running on multiprogramming IoT devices from untrusted OSes. It uses ARM TrustZone technology, securing critical applications through a lightweight runtime system responsible for the communication between the applications and the OS running in the normal world. This runtime system does not provide system services, but forwards the requests to the untrusted normal world, verifying the responses and employing a page based encryption mechanism to protect all the data segments from a security-critical application. Whenever an encrypted page is accessed, it is decrypted in the internal RAM, which is immune to physical exploits. It is not necessary for any modification to legacy applications. The authors tested the proposed solution with microbenchmarks and real applications, which presented negligible and, in a few cases, moderate overhead when running real applications.

3.1.4 Verification with Coloured Petri Net Models and Model Checking

Rodriguez et al. [88] presented a formal and executable model for the MQTT protocol, using CPN, and performed a model checking of behavioral properties. To alleviate the effect of the state explosion problem, the authors used a method called sweep-line. The authors verified properties regarding the MQTT connection and disconnection and publishing/subscribing processes. They implemented the sweep-line method and performed some experiments, concluding that even though their implementation takes more time to execute, it significantly reduces the amount of memory usage.

Villani et al. [112] proposed two hybrid processes to integrate model-based testing and model checking for the design of industrial systems. The authors also presented a tool supporting both proposed processes, named CONTEA, and presented 3 case studies to evaluate the hybrid processes. The evaluation concluded that the proposed processes have vantages when compared to only model checking or testing process, anticipating the detection and correction of possible problems in the design lifecycle.

Kunnappilly et al. [57] performed model checking to validate their proposed assisted living system architecture, which is composed of sensors collecting data, local and cloud

processing, and an intelligent decision support system. The authors specified the architecture using the Architecture Analysis and Design Language and described the architecture semantics with stochastic timed automata. The model checking ensured that the required functional behavior was met.

Wang et al. [116] presented a CPN model for model-based testing of the Paxos distributed consensus protocol. The CPN model, used for test case generation, was automatically validated through finite-state model checking regarding small configurations' correctness. The authors implemented the Paxos protocol with the Gorums framework, and the experiments with their testing approach demonstrated good code coverage considering unit and system tests. They conclude that their proposal can detect errors and bugs in the Paxos implementation.

Suriadi et al. [103] used a CPN model to represent a Privacy Enhancing Protocol, named Private Information Escrow Bound to Multiple Conditions Protocol (PIEMCP). A model checking was performed to validate some defined privacy compliance properties, using ASK-CTL and analyzing the generated state space. Seifi et al. [93] also applied a CPN to model and verify the OSAP protocol (Object Specific Authorization Protocol), which is responsible for defining how entities can access the protected objects in a TPM (Trusted Platform Module). Through model checking, the authors conclude that the authentication property of OSAP can be violated.

Attia [10] modeled and verified access control policies using CPN and performing model checking. Extensions to RBAC (Role-Based Access Control) and ABAC (Attribute-Based Access Control) models were proposed, and the author carried out an empirical study comparing the proposed model with three other models. The results demonstrate that the proposed model is more suitable for all the compared models. The following properties were verified through model checking: reachability, availability, role-role containment, weakest precondition, dead roles, and information flow.

Jaidka et al. [49] applied CPN to model safety-critical interactive systems. A simple infusion pump was modeled and verified through the state space analysis method, investigating the system's behavior. Z. Liu and J. Liu [68] modeled blockchain smart contracts using hierarchical CPN, considering some possible attacker models, and executed simulations to validate their functional correctness, performing model checking with ASK-CTL

branch time logic to detect latent vulnerabilities. Akhtar et al. [3] also used a hierarchical CPN to model a Flood Monitoring, Prediction, and Rescue (FMPR) system, validating the correctness properties of liveness and safety through rigorous model checking.

Xu et al. [123] used a Control Logical Petri Net to model critical operations of an automobile system. Authors performed experiments to compare the Statistical Model Checking (SMC) and Importance Sampling (IS) techniques, concluding that the IS seems to consume less time while providing accurate results equivalent to SMC. Yankson et al. [129] modeled a Smart Connected Toy system with a CPN considering privacy preservation elements to simulate the communication flow. Fremantle and Aziz [38] formally modeled the OAuthing personal cloud-based IoT system, which is responsible for performing authentication and authorization operations. Although authors did not use CPN modeling, they carried out model checking through the defined model, verifying that it meets the specifications and proving that security and privacy are preserved since certain communications do not occur, and data can only be shared safely according to user's consent.

3.2 State of the Art: TEEs for IoT Applications

In this Section, we present the state of the art of this research based on a systematic literature review (SLR). The following subsections present the protocol used to perform the SLR and the results achieved after the review and data extraction processes. The aim of this SLR is the use of TEEs to protect data in IoT applications.

3.2.1 Systematic Literature Review Protocol

3.2.1.1 Problem

As already discussed in Chapter 1, security is still a challenge for the Internet of Things applications, mainly due to the lack of well established and standardized mechanisms that really secure the generated data and also due to the constrained resources of the devices, such as limited processing capabilities, limited battery capacity, among others. To improve the confidence in general computer systems, the concept of "trusted computing" was created by the Trusted Computing Group (TCG), in the early 2000s. This is implemented using a

Trusted Platform Module (TPM), a special chip that provides features to secure the operating system and the application, and is supposed to be tamper-proof. In this context, there are the Trusted Execution Environments (TEEs), which have a drastically reduced Trusted Computing Base (TCB) and allow the creation of protected regions of memory through the use of an extension of the processor instructions set. It is interesting, then, to investigate how TEE technologies are being used to improve the security in IoT systems.

3.2.1.2 Research Questions

The research questions considered for the SLR are listed below:

1. What kind of IoT solutions TEE has been used for?
2. What kind of IoT scenarios TEE has been used in?
3. What are the advantages and disadvantages of TEE usage?
4. What have the authors proposed for future work?

3.2.1.3 Search strategy

To perform the search, we chose common keywords related to this study and the best synonyms of each keyword. To delimit the range of related works better, we applied the Population, Intervention, Context, and Outcome (PICO) approach. To perform the search, we considered the main available research repositories that include papers from Computer Science conferences and journals. The starting keywords and the terms considered to define the search string are listed below.

Keywords: Trusted Execution Environment, Internet of Things, Security

Population : Internet of Things, Web of Things, Edge of Things, Internet of Everything

Intervention : Trusted Execution Environment, Trusted Execution Technology

Context : Security, Privacy, Confidentiality, Integrity, Trustworthiness, Protection

Outcome : Solution, Technique, Tool, Approach, Method, Mechanism, Advantage, Benefit, Positive Point, Disadvantage, Drawback, Negative Point

To improve the number of keywords to be used in the source selection criteria, some papers were analyzed regarding title, abstract, and keywords. The selection of these papers was carried out by collecting the first result of a search in the following research repositories: ACM Digital Library, IEEE Xplore Digital Library, Scopus, Springer Link, ScienceDirect and Google Scholar. For this search, the following keywords were used as search string: "Internet of Things" AND "Trusted Execution Environment" AND "Security". The returned papers were then ordered by default according to its relevance, with the most relevant first. The first paper at the resulted list for each search is listed in Table 3.1, composing a total of six papers.

Table 3.1: First search results

Research repository	First paper in the result
ACM Digital Library (5)	A Secure, Privacy-Preserving IoT Middleware using Intel SGX
IEEE Xplore Digital Library (6)	IIoTEED: An Enhanced, Trusted Execution Environment for Industrial IoT Edge Devices
Scopus (61)	Fog Orchestration for the Internet of Everything: State-of-the-Art and Research Challenges
Springer Link (40)	MIPE: a Practical Memory Integrity Protection Method in a Trusted Execution Environment
ScienceDirect (23)	A Security Authorization Scheme for Smart Home Internet of Things Devices
Google Scholar (703)	A Survey of Fog Computing: Concepts, Applications and Issues

Since the first result in the ACM returned list is not a scientific paper (Hardware-Assisted Security: Promises, Pitfalls, and Opportunities), the second result was chosen (as listed in the Table 3.1). After this phase, some other terms were considered to compose the search string, as listed in Table 3.2.

Table 3.2: Terms considered for the search string

Keyterms	Alternative terms
Trusted Execution Environment	Trusted Execution Technology, Trustzone, Intel SGX, SGX, Software Guard Extensions, TEE, Keystone Enclave, Sanctum
Internet of Things	Edge Computing, Edge of Things, Embedded Systems, EoT, Internet of Everything, IoE, IoT, Web of Things, WoT
Security	Confidentiality, Integrity, Trustworthiness, Privacy, Protection
Solution	Approach, Framework, Mechanism, Method, Technique, Tool, Advantage, Benefit, Positive Point, Disadvantage, Drawback, Negative Point

3.2.1.4 Search String

The search string defined here is composed of keywords, alternative terms and the PICO terms. To link these terms, resulting in a representative search string, the boolean operators AND and OR were used. The established search string is listed below:

> (“Internet of Thing” OR “Edge Computing” OR “Edge of Things” OR “Embedded Systems” OR “EoT” OR “Internet of Everything” OR “IoE” OR “IoT” OR “Web of Things” OR “WoT”) AND (“Trusted Execution Environment” OR “Intel SGX” OR “Keystone Enclave” OR “Sanctum” OR “SGX” OR “Software Guard Extensions” OR “TEE” OR “Trusted Execution Technology” OR “Trustzone”) AND (“Advantage” OR “Benefit” OR “Positive Point” OR “Disadvantage” OR “Drawback” OR “Negative Point” OR “Security” OR “Confidentiality” OR “Integrity” OR “Privacy” OR “Protection” OR “Trustworthiness” OR “Solution” OR "Approach” OR “Framework” OR “Mechanism” OR “Method” OR “Technique” OR “Tool”)

After determining the search string, to validate its quality, we performed a search analyzing if six known papers were found with it. It is possible to see, in Table 3.3, that the search string was capable of finding all the expected papers.

Table 3.3: Validation of the search string

Papers	Result
T2Droid: A Trustzone-based Dynamic Analyser for Android Applications	Ok
TrustShadow: Secure Execution of Unmodified Applications with ARM TrustZone	Ok
IIoTEED: An Enhanced, Trusted Execution Environment for Industrial IoT Edge Devices	Ok
A Security and Trust Framework for Virtualized Networks and Software-defined Networking	Ok
Secure and Trusted Application Execution on Embedded Devices	Ok
MIPE: a Practical Memory Integrity Protection Method in a Trusted Execution Environment	Ok

3.2.1.5 Search Repositories

To expand the possibilities of finding good works, covering a high quantity of research sources, we decided to use the following scientific repositories (Table 3.4) that gather publi-

cations from many important conferences and journals:

Table 3.4: Selected scientific repositories

Scientific Repository	URL
ACM Digital Library	http://portal.acm.org
El Compendex	http://www.engineeringvillage.com
IEEE Digital Library	http://ieeexplore.ieee.org
Wiley Online Library	https://onlinelibrary.wiley.com/
Scopus	http://www.scopus.com
Springer Link	http://link.springer.com

As ScienceDirect and Scopus are both from Elsevier and the first has strict limitations regarding the search string (characters and boolean operators quantity), we decided to consider just the Scopus base. For the same reason, Google Scholar was excluded from our previous list of research indexers since it also limits the size of the search string, making it difficult to use the defined search string. The formatted search string and the number of documents resulting in each scientific repository are presented in Appendix A.

3.2.1.6 Selection Criteria

To refine the search, avoid results that can not help answer the research questions, and improve the probability of obtaining good results, we established the selection criteria for the exclusion and inclusion of papers.

Exclusion Criteria

- Posters, short papers and extended abstracts (papers with less than 3 pages);
- Book chapters, PhD and Master theses;
- Papers not written in English;
- Papers that do not focus on TEE usage for IoT security;
- Duplicate results;

- Papers published before 2000 (trusted computing was defined in earlies 2000);
- Secondary studies.

Inclusion Criteria

- Journal and conference articles;
- Results that answer the research questions.

3.2.1.7 Selection Procedure

To select the interesting documents for this SLR, we used the selection criteria and followed these steps:

1. Exclude duplicate documents;
2. Exclude documents published before 2000 or documents not written in English;
3. Exclude documents that were not published in conferences or journals;
4. Exclude book chapters, PhD and Master theses, extended abstracts, short papers, posters, secondary studies (e.g. surveys and reviews);
5. Exclude irrelevant documents, i.e. documents that are not related to TEE usage for IoT security.

To exclude irrelevant documents, after we performed the exclusions described in the four initial points, each entry of the resulted list was analyzed by two reviewers according to the following criteria:

- Each reviewer classified the document as relevant, undefined or irrelevant;
- Documents classified as relevant by two reviewers are maintained;
- Documents classified as irrelevant by two reviewers are excluded;
- Documents classified as undefined by two reviewers are better analyzed, through a fast reading of the entire document, and then reclassified as relevant or irrelevant;

- Documents classified as relevant or undefined by one reviewer and as irrelevant by another reviewer are discussed between both until they agree regarding one of the previous classifications.

3.2.1.8 Quality Assesment

To assess the papers quality, we elaborated the following list of questions:

1. Is the text well organized and clear (easy to understand)?
2. Is the motivation and objective well described?
3. Does the paper present an application scenario or case study?
4. Is the methodology clear (easy to understand and replicate)?
5. Does the paper have many references and good related works?
6. Does the study present some implementation and practical results?
7. Does the paper clearly present any advantages/disadvantages regarding the technology used?
8. Do the authors present good validation?
9. Are the results clear enough (explicit and well discussed/evaluated)?
10. Do the authors suggest future works?

For each selected paper, the reviewers answered each of those questions with three possible answers: yes, partially or no. The paper quality was assessed according to the score attributed for each answer:

- Yes - 1;
- Moderated - 0.5;
- No - 0.

Thus, since there are 10 questions, the maximum score can be 10 whereas the minimum score can be 0, indicating, respectively, the highest and the lowest quality for a paper. Depending on the papers score, they were classified as:

- High quality, if scored above 6;
- Medium quality, if scored between 4 and 6;
- Low quality, if scored below 4.

To answer each of the quality assessment questions, we elaborated the following guideline, enumerated according to the question number:

1. Only receives 1 if the overall text is really well written, organized and clear;
2. Only receives 1 if the motivation and the objectives are clear and well explained;
3. Only receives 1 if the paper clearly mentions an application scenario (e.g., smart home application, payment system, etc.);
4. Only receives 1 if the methodology is clear, involving the experimental design, experiment execution and results definition;
5. Only receives 1 if the paper is well grounded, with good related references;
6. Only receives 1 if there is implementation for the proposal and practical results well described;
7. Only receives 1 if there are advantages and disadvantages about the use of TEE;
8. Only receives 1 if the authors present a good validation for the proposal, practical or formal, well described and grounded;
9. Only receives 1 if the results are clear, well discussed and well evaluated;
10. Only receives 1 if the authors suggest future works related to their proposals.

3.2.1.9 Data Extraction

The data that should be extracted by the reviewers are listed below:

- Title;
- Authors;
- Abstract;
- Year;
- Type of article;
- Conference/Journal name;
- Country/countries where the research was carried out;
- Number of pages;
- Number of citations;
- Quality;
- IoT solution;
- IoT scenario;
- Advantages and disadvantages of TEE usage;
- Future work suggestions.

3.2.1.10 Execution

After performing the search in all the selected scientific repositories, in August 2018, we got a list of 541 results. Each entry of this list was judged and classified, by four reviewers, as relevant, irrelevant or undefined. After this process, we got:

- 134 undefined papers;
- 43 relevant papers;

- 364 irrelevant papers.

We then started a new review process, dividing the 134 undefined papers into two groups of 67 papers for each pair of reviewers. The reviewers performed a better analysis of each undefined paper's relevance and reclassified them as irrelevant or relevant. In the end, we selected a total of 58 relevant papers for the SLR. The list of the selected papers can be found in Appendix A (Section A.1).

3.2.2 Systematic Literature Review Results

In this Subsection, we show the collected information regarding each of the four research questions, i.e., IoT solutions, IoT scenarios, advantages/disadvantages, and future works.

3.2.2.1 IoT Solutions

All the IoT solutions approached in the selected papers can be classified in one of the following solution types:

- Application - considering all the works that propose security solutions to specific applications, such as protection for video applications or framework to develop secure solutions;
- OS (Operating System) - considering all the works that propose security solutions related to OS, such as protection for application execution or memory;
- Security - considering all the works that propose security mechanisms as solutions, such as authentication or attestation methods.

In Figure 3.1, we can see the proportion of the number of works classified according to the solution types. As observed, the number of works addressing the three types of solutions is well distributed, presenting similar proportions. We can see a difference of just six papers between the greater bar (OS solutions) and the smaller one (Security solutions). The number of works addressing OS solutions is slightly higher than the number of works addressing Application solutions, whereas the latter is slightly higher than the number of works addressing Security solutions.

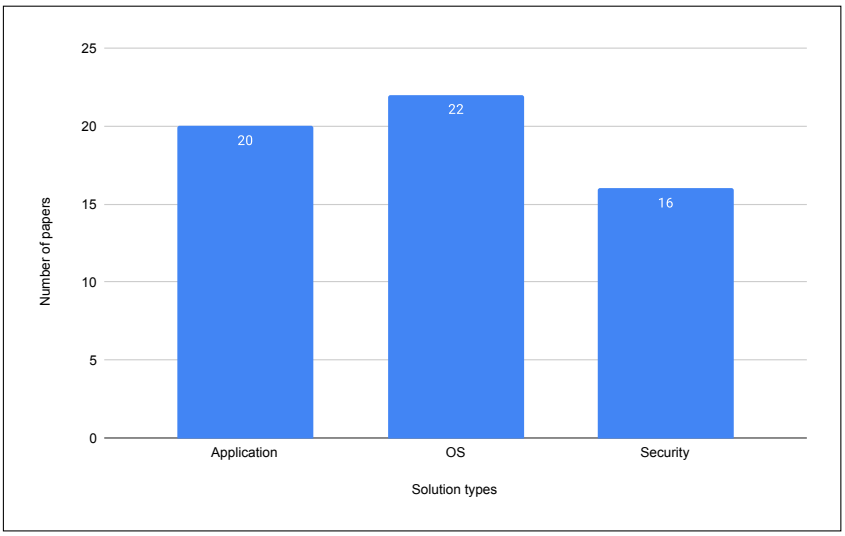


Figure 3.1: Solution types

Among the works classified as OS, we found five types of solutions: hypervisor, application execution, communication, memory dump, and access control. As we can see in Figure 3.2, the majority of OS solutions propose protection for the execution of the application. We list all the OS solution types in the Table 3.5.

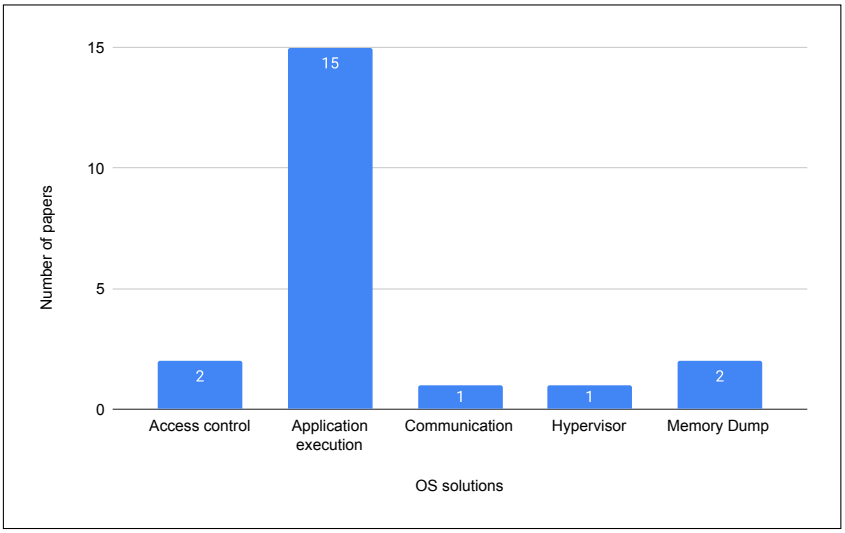


Figure 3.2: OS solutions

For the works classified as Application, we found six types of solutions: system, framework, architecture, platform, communications checker, and comparison with other solutions (multi-party computation). In the Figure 3.3, we can see that most of the Application solutions propose some system to protect data. All the application solution types can be found in

Table 3.5: OS solution proposals

OS solutions	
Protection for legacy applications [41]	Access control between trusted and untrusted worlds [127]
Protection for embedded devices [69]	Hardware isolation architecture [33]
Protection for applications execution [20]	General-purpose trusted computing platform [36]
Protection for real-time OS [81]	Trusted OS [128]
Protection against data leakage [17]	Memory dump mechanism [102]
Protection for memory integrity [21]	Cache-assisted secure execution [131]
Protection for legacy applications [42]	Runtime security architecture [19]
Protection for services [85]	Kernel protection mechanism [135]
Protection for SoC communication bus [99]	TCB measurements management [77]
Trusted I/O [79]	Hypervisor [82]
Secure device access method [54]	Secure mobile device framework [62]

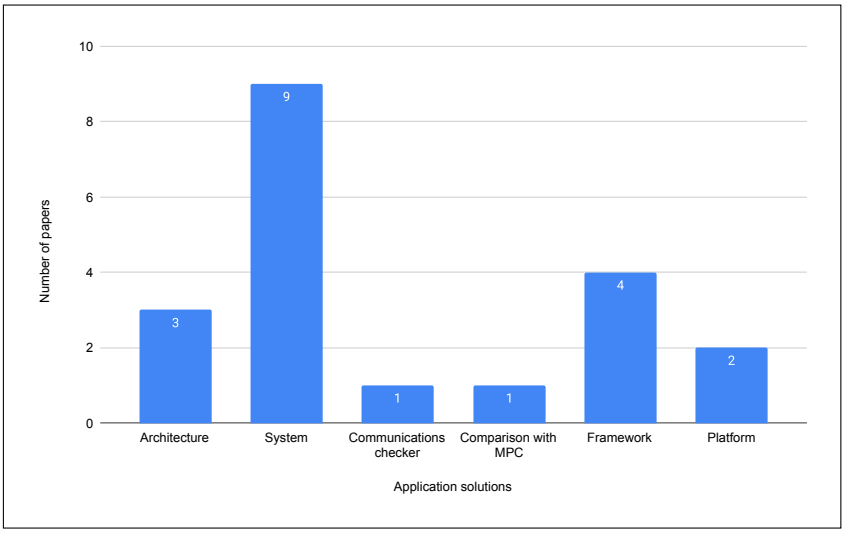


Figure 3.3: Application solutions

the Table 3.6.

Lastly, we found six types of solutions for the works classified as Security: authentication, attestation, authenticity, keys derivation, keys distribution, and rootkit. We can see in Figure 3.4 that authentication and attestation proposals are prevailing in the Security solutions. We show all the security solution types in the Table 3.7.

Table 3.6: Application solution proposals

Application solutions	
Protection for flow tracking application [114]	Trusted framework to develop IoT applications [124]
Protection for health data [66]	Secure architecture for P2P scenarios [76]
Protection for edge computing [80]	Comparison among TEE and secure MPC application [7]
Protection for video application [37; 126]	Secure logger [95; 74]
Protection for system analyser [125]	Societal model for IoT security [108]
Protection for location-based services [56]	Trusted auditor [72]
Protection for data dissemination [91]	Data encryption mechanism [122]
Protection for data management [12]	Data protection (app) [132]
Protection for data aggregation (app) [100]	Checker for Industrial gateway communications [84]

Table 3.7: Security solution proposals

Security solutions	
Lightweithg anonymous authentication [113]	Remote attestation mechanism [58]
Device snapshot authentication system [64]	Control-flow attestation [1]
Authentication scheme [53]	Remote attestation and channel protection [96]
Secure authentication and key distribution [61]	Boot attestation [92]
Protection for data through authentication [134]	Protection and attestation for remote terminal [115]
Device private keys protection architecture [78]	Remote attestation [2]
Keys derivation from device characteristics [63]	Authenticity detection service [34]
Keys protection against cold boot attacks [133]	Cache rootkit exploiting TrustZone [130]

3.2.3 IoT Scenarios

Among all the selected papers, approximately only 24% presented IoT scenarios. All the identified scenarios are listed below:

- Automotive - device access for automotive software [54] and secure communication between vehicle infotainment system and user devices [61];
- Healthcare - secure heartbeat sensor application [124], critical medical services [85] and healthcare data monitoring system [66];

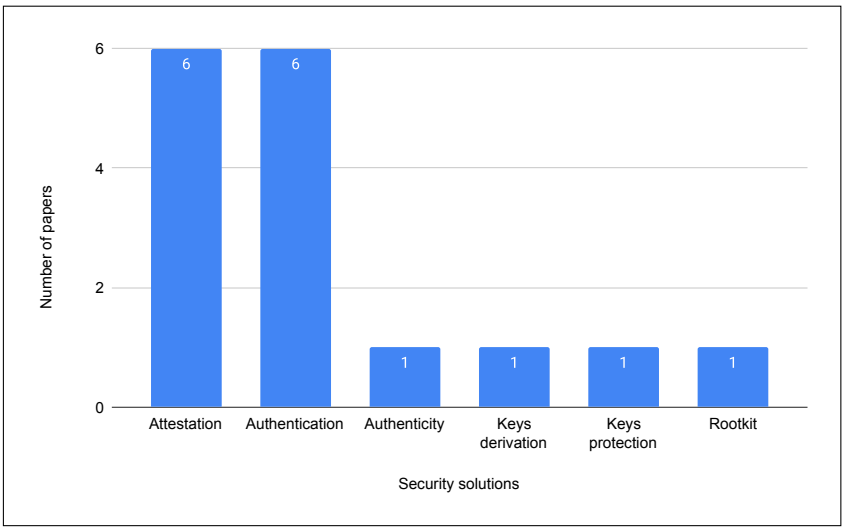


Figure 3.4: Security solutions

- Smart metering - data dissemination [91] and data aggregation [100];
- Video application - video surveillance devices [37] and duplicate removal operations [126];
- Edge computing - industrial IoT edge devices [81] and edge-cloud communications [80];
- Industrial maintenance - secure smart service for industrial maintenance scenarios [64];
- Payment system - trusted OS for mobile payment system [128];
- Smart home - device authentication for smart home/smart grid scenario [78].

The rest of the papers (about 76%) did not present any IoT scenario. These works are related to embedded systems or mobile devices/smartphones solutions.

3.2.4 TEE Advantages and Disadvantages

Only 41% of the selected papers presented any advantage or disadvantage regarding the adoption of TEEs. The TEE vendors (e.g., ARM and Intel) already present many of the general advantages, such as hardware isolation (normal world and the secure world) and memory protection [128; 64; 19; 77; 95; 74; 12; 79; 100; 41]. This helps to prevent malicious

applications and to isolate sensitive data [19], and enables to protect code running in the secure world [34; 41] as well as to protect against software attacks [95]. Another considered advantages are low performance overhead [37; 42; 7; 62; 134; 72; 114; 56; 91; 115], low power consumption overhead [37] and low latency overhead [62].

One mentioned disadvantage is the well-known fact that TEE is not resistant to side-channel attacks [64; 62]. According to Zhang et al. [130], the incoherence between cache in the normal and secure worlds is a disadvantage related to TrustZone, since someone can explore it as a vulnerability. Although some works presented low overheads, some authors consider that some operations with TEEs present high-performance overhead [125; 96], e.g., when using the SGX monotonic counter [74], or high power consumption overhead [72]. For others, the need to use specific hardware is also considered a disadvantage [100].

3.2.5 Suggested Future Works

Only 48% of the selected papers presented suggestions for future work. In general, all the suggestions relate to improvements or evaluations of the proposals, integration with other mechanisms, or comparison with other solutions.

Osterhues et al. [76] suggested integrating the secure architecture for P2P scenarios with the OpenMoko project. For the general-purpose trusted computing platform, Feng et al. [36] suggested implementing it in a specific board, improving the prototype, adding a TPM 2.0 module, and implementing Trusted Applications. Yang et al. proposed to improve and evaluate their Trusted OS [128]. Lesjak et al. [64] recommended comparing other solutions with their device snapshot authentication system.

Kylanpaa et al. [58] suggested implementing their remote attestation mechanism using Qemu and a real TrustZone to modify the TEE initialization and change the TA loading to the bootloader phase, to extend the measurement mechanism and to improve the keys storage or add a Public Key Infrastructure. Zhang et al. [130] proposed to test their cache rootkit in architectures different from TrustZone and improve the defense mechanisms.

Pinto et al. [82] suggested, for their proposed hypervisor (LTZVisor), to evaluate real-time aspects with short-term and long-term tests, to investigate timing interferences and sources of non-determinism, and to extend the solution for new platforms. Zhang et al. [133] recommended extending the keys' protection against cold boot attacks to store multiple keys

and ensure a parallel encryption process. Park and Kim [77] suggested extending the TCB measurements management system and applying SGX to the miners.

For the secure logger solution, Shepherd et al. [95] proposed group logging schemes for multiple devices and comparing the TEE performance. Pinto et al. [81] proposed implementing the protection for real-time OS in edge devices and integrating it with other hardware anchors. For memory integrity protection, Chang et al. [21] suggested improving its algorithms and process, implementing it with the B method, and formalizing it. Liang et al. [66] proposed integrating the protection for health data with a blockchain-based access control scheme.

Guan et al. [42] proposed introducing another level of virtual memory to their solution, protection for legacy applications, and integrating it with existing shielding mechanisms. Yalew et al. [34] indicated to optimize their authenticity detection service. Zhao et al. [134] suggested providing secure communication for their authentication-based data protection solution. Qin et al [84] recommended adding security enforcement policy for their industrial gateway communications checker. Kulkarni et al. [56] proposed to evaluate their solution, a protection system for location-based services, and compare it with other approaches.

For the boot attestation process, Schulz et al. [92] suggested supporting additional device types, establishing trusted channels, and extending and refining the protocol. For the remote attestation and channel protection solution, Shepherd et al. [96] proposed to establish a trusted channel shared between multiple devices, to extend the performance evaluation, and to reduce critical sizes and computational overhead with ECDH cryptography. Ayoade et al. [12] suggesting adding private blockchains for their data management protection system. Cao et al. [17], for their mechanism against data leakage, proposed to adjust the size of the sliding window for individual processes and to find a smarter page replacement algorithm.

Silva et al. [100] recommended combining different approaches to strengthen the security and privacy of their data aggregation system. Siddiqui et al. [99] proposed to enable secure configuration and to provide hardware resources segregation in their SoC communication bus. Raes et al. [85] suggested exploring use cases opportunities for their services protection solution. Peters et al. [79] recommended extending their trusted Bluetooth I/O to address other I/O paths (e.g., WiFi and NFC) and evaluating its performance cost. Lee et al. [61] suggested adding a new authentication method using blockchain for their secure

authentication and key distribution system. Guan et al. [41] proposed improving the signing mechanism of their solution, protecting legacy applications, and verifying the authenticity of a manifest.

3.2.6 Threats to Validity

To deal with the internal validity, i.e., get confidence for the whole performed process, we adopted a peer review process, minimizing the subjective bias of each reviewer, since pairs of reviewers carried out the overall selection process. Although each reviewer's knowledge is distinct from the others, each pair of reviewers had a reviewer with more knowledge on the study topic. Whenever doubts arose during the selection process, both reviewers achieved a consensus regarding the decision to include or exclude a paper. Besides, we think the defined search string is comprehensive enough to comprise the central primary studies to extract the answers for the research questions.

To minimize possible problems with the external validity, i.e., the generalization of the found results, we decided not to limit the search for specific journals and conferences. This way, we included papers from any conferences and journals, avoiding the removal from primary studies just because of the publication venue. Furthermore, we also avoided excluding papers with a low score, since the score could be skewed according to the reviewers' subjective bias, even following the quality assessment guidelines.

Related to conclusion validity, and to avoid bias, for each selected paper, a reviewer checked the data collected by another one. This way, if the proposed protocol is applied, following these considerations to replicate this study, we are confident that the same results can be achieved.

Lastly, since the focus of this study is the Internet of Things applications, we could restrict the search string to involve only the TrustZone technology, since it is the only TEE available in the market that comes in IoT devices, due to the ARM processor. If we did this, we would be excluding all the solutions that consider the Intel SGX to improve security in edge/fog/cloud-based IoT scenarios. This way, to avoid problems with the construct validity, we elaborated the search string to include all TEE solutions applied to IoT applications in distinct scenarios (edge, fog, and cloud).

3.2.7 Challenges and Directions

Next, we present a discussion with respect to challenges and directions regarding the use of TEEs considering the following topics: *vulnerabilities*, *development complexity*, *remote attestation*, *communication channel* and *solutions integration*.

Concerning *vulnerabilities*, one of the main challenges regarding the adoption of TEEs, regardless of the application scenario, is the known vulnerability to side-channel attacks. Although this kind of attack is considered a bit complex to execute, demanding a specialized level of technical knowledge, we should always consider it as a possible threat. Given this problem, technical artifices can be proposed to mitigate or avoid this threat. Besides, concerns remain with code running inside a trusted application, since it may contain vulnerabilities that unauthorized parties can explore to compromise the TEE system. Thus, developers must continue following the recommendations to secure code, avoiding, for instance, buffer overflows, race conditions, and uninitialized variables. This can also be a research topic to be explored.

In the context of the *development complexity*, another known challenge is the learning curve needed to develop trusted applications with either TrustZone or SGX. Efforts, in this sense, can also be applied to ease the development of trusted applications. For example, Scone [107] facilitates the deployment of SGX applications through containers, Python SGX [14] provided means to develop SGX applications using Python (it is not maintained anymore) and Rust OP-TEE TrustZone SDK [105] enables the development of TrustZone applications using Rust.

For *remote attestation*, applications running inside a TEE should provide a means that attests their trustworthiness to any interested third-party, i.e., that proves they are running inside real TEE hardware. For this, the SGX technology provides a remote attestation protocol in which the third-party application challenges the trusted application, supposedly running inside an SGX enclave. The trusted application replies with specific information from the enclave, allowing the third-party application to verify this content with the Intel Attestation Service (IAS). After a successful verification with the IAS, once the trustworthiness is proved, both parties can establish a secure communication channel, since they generate an ECDH (Elliptic-Curve Diffie-Helmann) shared key during the process. As the TrustZone

does not provide a specific and standardized development kit, like SGX, each Trusted OS should provide its remote attestation process. Thus, remote attestation is also a good topic for future researches since some Trusted OSes, such as OP-TEE [106], still do not provide such a mechanism.

The *communication* between the untrusted applications and the trusted applications, i.e., between the normal world running a Rich OS (e.g., Linux or Android) and the trusted world running a Trusted OS (e.g., OP-TEE or Kinibi [104]), can also be a target of attacks. To mitigate this, the platforms should implement secure communication giving special attention to the shared memory between both worlds. This extends the possibilities for new research works. In this sense, the platforms must avoid that unauthorized processes running inside the normal world can access information running inside the trusted world.

Considering the *solutions integration*, we can notice that TEE is being applied together with other security solutions and mechanisms. For instance, some works that apply TEE to protect data also apply blockchain [77; 66; 12; 132]. In general, the data are encrypted to be processed only inside a TEE application, being protected in rest or transit. Thus, the data hashes can be stored in the blockchain for auditing operations, verifying the data integrity once the trusted applications process them.

3.3 Summary

In this Chapter, we saw that data security is an open challenge for IoT applications and many works have addressed solutions in the sense of mitigating this problem. We firstly presented general works describing the main challenges and proposed solutions. Then, we presented works that apply TEE to protect data in IoT applications and works that apply CPN and model checking to validate their proposals. Lastly, we presented our SLR to get an overview of state of the art regarding TEE's use for IoT applications, since our proposal contributes to improving data protection through trusted computing. We described the review protocol and presented the achieved results after the analyses of the 58 relevant selected papers.

Although considerable effort has been employed to provide data security solutions for IoT applications, it is clear that there is still a lack of standardized means to protect sensitive data, and this is was our motivation to propose our work.

Chapter 4

Trusted Architecture for IoT

This Chapter describes the scenario considered as the basis for this work, the key security principles applied, the threat model adopted, and the trusted architecture proposed. Besides, it also describes two trusted components proposed for the architecture: the Key Vault and the Trusted PEP Proxy (TruPP).

4.1 Base Case Scenario

The base case scenario considers a residential environment in which residences have smart meters to measure energy consumption for billing purposes. A basic architecture for this application includes the following entities: smart meters, a broker acting as a publish/subscribe system, and a consumer application responsible for processing the data.

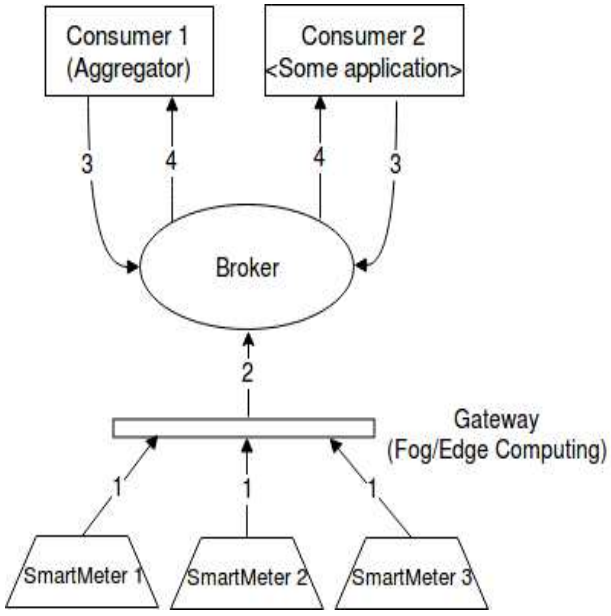
In this scenario, the smart meters are the data producers, and the processing applications are the consumers. Figure 4.1 shows this typical architecture. The communication flow between components can be as follows:

1. The Producers (Smart Meters) send energy consumption data to the IoT Gateway;
2. The IoT Gateway preprocesses these energy consumption data and sends them to a Broker (Publish/Subscribe System);
3. The Consumers register interest on the specific Producers, in the Broker, to receive notifications about their energy consumptions;

4. The Broker sends notifications with the Producers' energy consumption data to the Consumers that previously registered interests.

Such IoT Gateway can be any device more powerful than the smart meters, for example, capable of processing more complex operations on their generated data or having better communication capabilities. This scenario is often known as fog/edge computing, as mentioned in Chapter 1, since the IoT devices connect to another device with more processing power, which is closer to the network edge [43].

Figure 4.1: Typical architecture for a smart metering application.



4.1.1 Aggregation of Smart Meter Data

For this case scenario, the consumers are data aggregators that process the energy consumption data for billing purposes. Such data aggregation is made by summing all the energy consumption data collected for each residence in a specific region. To perform this aggregation, it is necessary to read all the energy consumption data from all the residences in a region.

As stated in the Introduction (Sections 1.1 and 1.2), there are non-invasive approaches to estimate what kind of electronic devices or household items are used at a specific time and for how long. One of these approaches is called NIALM (Non-Intrusive Appliance Load

Monitoring) and can be considered as a privacy threat since an evil-minded adversary can discover what people are doing in a specific residence (e.g., an attacker can get information about the time in which people have a shower, watch TV or use the microwave).

4.1.2 Need for data protection

Due to the privacy problem previously mentioned, energy consumption data are considered sensitive, needing protection. Thus, it is necessary to apply some mechanisms to protect the data generated by the producers, avoiding the discovery of the users' behavior in the monitored residences. The architecture proposed in this work to protect these sensitive data uses some techniques, mechanisms, and the proposed component for keys management named Key Vault, which is described later (Section 4.4). For this, the architecture considers authentication for the entities, access control for the protected services, and cryptography for the sensitive data.

4.2 Threat Model

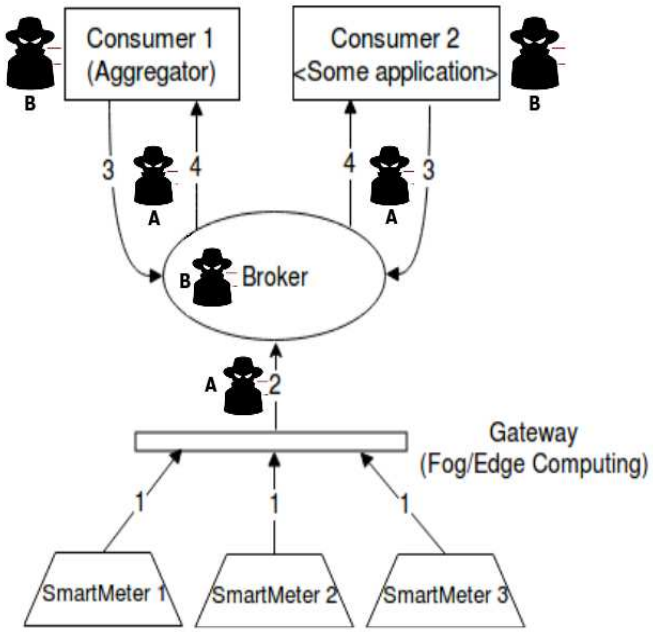
The study application is composed mainly of three parts: the producers (smart meters), a publish/subscribe system, and the consumers (data aggregators). The attack surface for this application, presented in Figure 4.2, considers:

- the possibility of an attacker gaining access while the data are in transit, transmitted between the parts, represented by the letter A;
- and the possibility of an attacker gaining complete access to the publish/subscribe system and to the consumers' hosts, represented by the letter B.

An attacker can intercept the messages sent from the producers (smart meters) to the publish/subscribe system or even the notifications sent from the publish/subscribe system to the consumers (aggregator). Besides, an attacker can get data having access to a consumer host or to the publish/subscribe system. In any case, the sensitive data are compromised. Then, the proposed architecture must deal with these threats, avoiding that any attacker in any of these situations can read sensitive data.

The solution considers that all the producers (smart meters) are secure, and does not deal with the possible existence of fake/crook producers. In any case, it is worth mentioning this possibility could cause only *some noise to generated data by valid producers, and this would not be a privacy problem for the users*.

Figure 4.2: Threat model for the base scenario



4.3 Principles of the Trusted Architecture

To protect the sensitive data, according to the specified threat model, we propose cryptography for data security and privacy, an identity management (IdM) system for entities authentication, and a simple access control system for the protected services. The proposed trusted architecture is based on the fact that the producers (source) encrypt all the generated data, before transmitting them to the publish/subscribe system, and only a trusted consumer can decrypt these data, once it is running on a trusted execution environment (TEE) and has access to the correct key. Furthermore, all the producers and consumers have identity credentials, such as X.509 certificates[1], and a PEP Proxy performs the access control using the

[1] `https://searchsecurity.techtarget.com/definition/X509-certificate`

OAuth2 protocol[2], which is an industry-standard for authorization.

Based on this architecture, the data encryption in the producers enforces a secure transmission of the data allowing only secure processing in a trusted machine, since the consumers must run on a TEE, such as Intel SGX or ARM TrustZone. Therefore, the users' privacy is preserved because the sensitive data are decrypted and processed only within a TEE application.

To allow the data encryption and decryption between entities without using a Public Key Infrastructure (PKI), we proposed a component for generating, storing and distributing the cryptographic keys used by producers and consumers: the Key Vault. It also runs on a TEE-enabled machine and is detailed next.

4.4 Key Vault

A Public Key Infrastructure (PKI) is needed to supply appropriated keys to the specific entities in a secure way to support the proposed solution. Producers must encrypt their data before sending them to the publish/subscribe system. Consumers must have the appropriate key to decrypt producers' data. Symmetric cryptography is not suitable for this scenario, because an adversary can get access to the producer credentials, be authenticated, and request the symmetric key. With this symmetric key, this adversary can decrypt all sensitive data from the producers, attacking the users' privacy.

To generate, maintain and distribute keys securely, we proposed the Key Vault, which runs in a trusted execution environment and provides public and private keys (asymmetric cryptography) accordingly and respectively to producers and consumers from our scenario. Thus, as the producers have access to public keys, if an adversary gets access to some producer credentials and gets authenticated, he/she can not decrypt any data sent to the consumer. In such situation, the adversary can only generate noise (i.e., he/she can generate fake data, encrypt with the public key and send them to the publish/subscribe system), which, as said before, is not a privacy concern. The communication with the Key Vault should be done through Transport Layer Security (TLS), using HTTPS, for instance. The Key Vault provides public keys for authenticated producers and private keys for authenticated and attested

[2] `https://oauth.net/2/`

consumers.

The Key Vault treats the requests from producers and consumers, after being authenticated, and sends the appropriate key (public for producers, private for consumers). As it runs in a TEE application, the keys are handled securely. The producers must attest the Key Vault before requesting public keys, to make sure the Key Vault is trustful, as well as the Key Vault must attest the consumers before sending private keys. Only TEE consumers can receive the private keys, after the remote attestation properly successful. The Key Vault encrypts the private key with a shared key that is generated or exchanged in the remote attestation process. If the remote attestation of the consumers fails, the private keys are not sent.

4.5 Trusted PEP Proxy

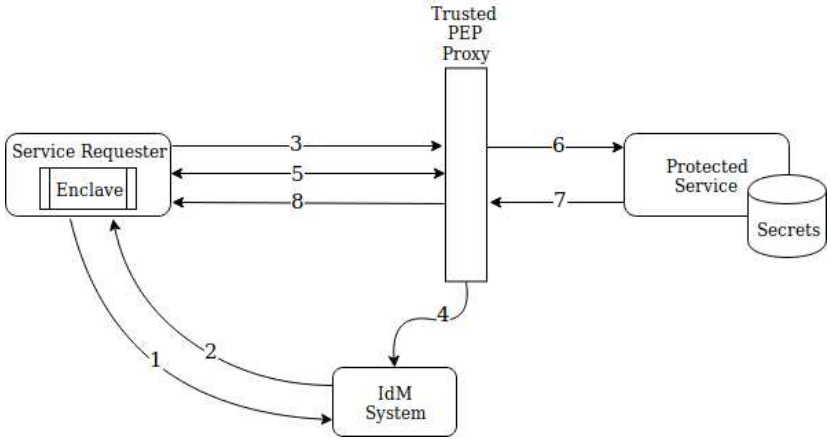
As seen before, a PEP Proxy intercepts requests to a protected service and checks if this request comes from a previously authenticated party, which should be authorized to get access to that service. The intercepted request usually contains a token, that should be validated with an Identity Management (IdM) system to check if the requester was authenticated and is authorized. In the positive case, the requester gets access to the protected service; otherwise, the PEP Proxy denies access, and the data service remains safe. [94; 15; 109]

Although the protected data service can be accessed only by authenticated and authorized requesters, there is no reason to trust their machines that process the data. For instance, an adversary can get access to their operating systems, with high-level privileges, and thus get access to sensitive data. This concern also creates a need to require that a service consumer runs on a TEE machine, and, this way, the trustworthiness can be improved.

Then, this work also proposes an improvement to a PEP Proxy in a way it can also perform the remote attestation process, to know if a requester runs a TEE application, besides only the authentication and authorization checking. Therefore, after authentication and authorization checking, the remote attestation process is initialized to verify if the requester application, which is interested in sensitive data, runs on a TEE and, accordingly, is considered a trusted entity. This modification in a PEP Proxy improves the trust in data consumers/requesters, once it allows the sensitive data distribution only to trusted entities, which

means that these entities were previously authenticated, authorized, and, now, attested. The architecture of this improved PEP, named here as Trusted PEP Proxy (TruPP), can be seen in Figure 4.3.

Figure 4.3: Architecture with the Trusted PEP Proxy



The communication flow between the components seen in Figure 4.3, which is enumerated, is described below:

1. the service requester, which is supposed to run a TEE application inside an Intel SGX enclave, authenticates with the IdM System, informing valid credentials;

2. after the authentication process, validating the service requester credentials, the IdM System returns a valid OAuth2 token, used later to check authorization (access permissions);

3. the service requester sends a request to the protected service, which is intercepted by the TruPP, informing the OAuth2 token received from the IdM System after the authentication process;

4. the TruPP verifies the OAuth2 token with the IdM System, checking if the service requester has the right permissions to access the protected service;

5. once the service requester has access permissions to the protected service, the TruPP starts the remote attestation process to verify if the service requester runs a TEE application;

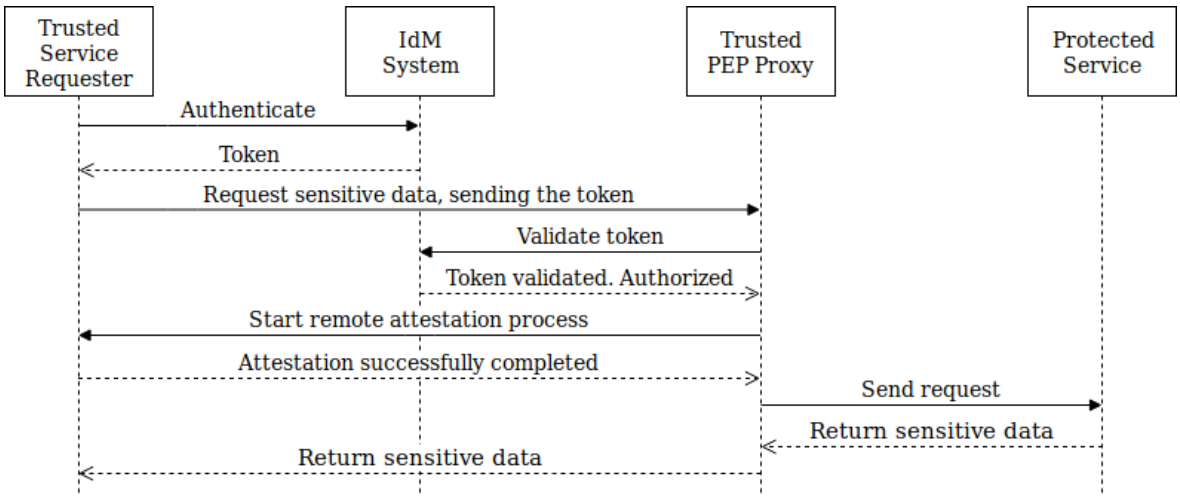
6. after the successful remote attestation, if it succeeded, the request reaches the protected service, which processes the return;

7. the protected service returns the requested data to the TruPP;

8. the TruPP forwards the requested data received from the protected service.

The main difference between a common PEP Proxy and the improved TruPP is step 5, which is the execution of the remote attestation process, checking for a TEE application in the service requester. The message sequence diagram of this communication flow can be seen in Figure 4.4. As explained, the trusted service requester authenticates in the IdM System and receives a valid OAuth2 token. Then, it sends a sensitive data request to the protected service, which is intercepted by the TruPP. The TruPP validates the OAuth2 token with the IdM System and starts the remote attestation process. Once the trusted service requester is successfully attested, the TruPP sends the request to the protected service and receives back the sensitive data requested. Finally, the TruPP forwards the sensitive data to the trusted service requester, which will securely perform the processing (inside an enclave).

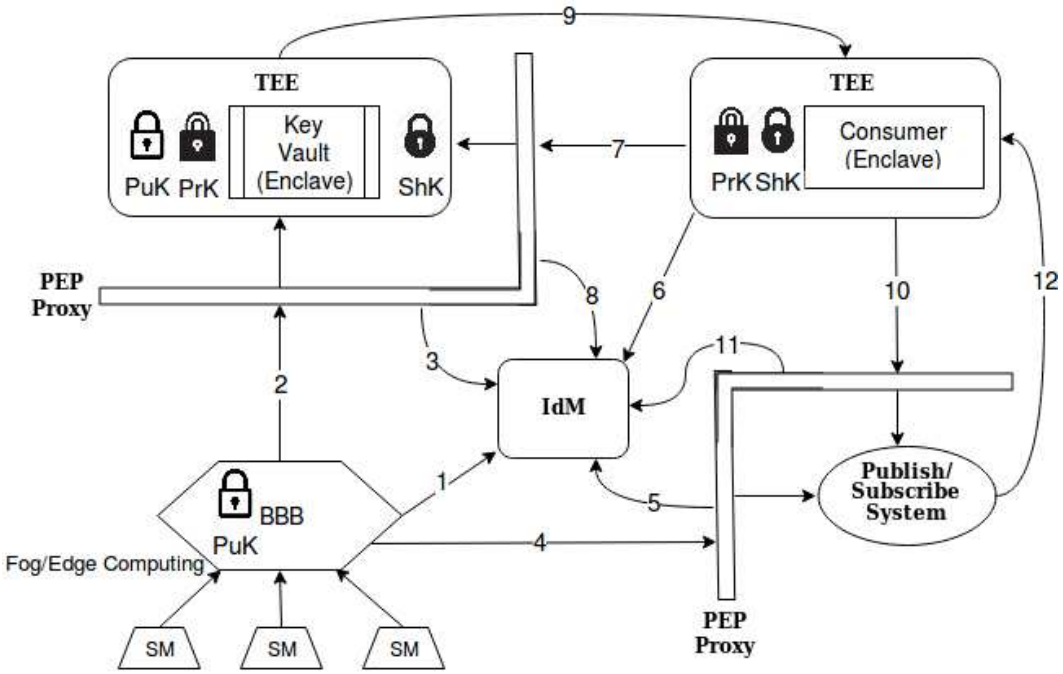
Figure 4.4: Message sequence diagram



4.6 Trusted Architecture

The trusted architecture proposed can be seen in Figure 4.5, and it comprises the following components:

Figure 4.5: Producer and Consumer Communication Flow.



- Producers - the smart meters, responsible for measuring the energy consumption, with or without an IoT gateway, or another device acting as a gateway, to process the generated data (Fog/Edge Computing) before sending them to a broker (e.g., publish/subscribe system). In Figure 4.5, they are represented as SM, for smart meters, and BBB, for BeagleBone Black acting as a gateway, but the producers can be any IoT device as well as the gateway can be another device;

- IdM System - the system responsible for Identity Management, which authenticates entities and provides, for instance, valid OAuth2 tokens for authorization purposes;

- PEP Proxy - the Policy Enforcement Point responsible for allowing/disallowing access to protected services after receiving an OAuth2 token and validates it with the IdM System. This architecture considers two instances of this component: one to protect the Key Vault and another to protect the Broker (Publish/Subscribe System);

- Broker (Publish/Subscribe System) - the context broker responsible for receiving the producers' data and sending them to consumers through notifications. Although we are using a publish/subscribe system as an example, the broker can be any other component that stores and transmits the data;

- Key Vault - the proposed system running on a TEE and responsible for securely managing, storing, and distributing the cryptographic keys to producers and consumers;
- Consumers - the data aggregators responsible for processing the energy consumption measurements for billing purposes. For the proposed architecture, a consumer should also run on a TEE.

The communication flow for producers and consumers is detailed next. As mentioned before, the producers, which are the smart meters, encrypt the data. This encryption is performed as a privacy technique to achieve data privacy. The consumer decrypts the sensitive data and processes them securely. All the steps in the communication flow are described below, and can be seen with a respective number in Figure 4.5 (1 to 5 for producers, and 6 to 13 for consumers):

1. The producer (smart meter or its gateway) accesses the IdM System passing valid credentials to be authenticated and get an OAuth2 valid token;
2. The producer attests the Key Vault to ensure that it executes on a TEE. Then, the producer sends to Key Vault a request asking for a public key and passing the OAuth2 token obtained with the IdM system, in the authentication process;
3. The PEP Proxy intercepts the request, checks the validity of the token with the IdM system, and forwards the request to the Key Vault if the token is valid. The Key Vault then sends back, to the producer, the asked public key (PuK in the Figure 4.5);
4. With the public key (PuK), the producer can encrypt the generated data (energy consumption) and send them to the Publish/Subscribe System;
5. Same as step 3 (PEP Proxy checks the validity of the token with the IdM system and forwards the request to the Publish/Subscribe system if the token is valid).
6. The consumer, a TEE application, accesses the IdM system passing valid credentials to be authenticated and also gets an OAuth2 valid token;
7. The consumer sends a request to the Key Vault asking for the private key (PrK in Figure 4.5) related to that public key sent to the producers and passing the OAuth2

token obtained with the IdM system. Besides the OAuth2 token, the request also informs an endpoint to enable the Key Vault to attest that the consumer is a real TEE application (remote attestation process);

8. The PEP Proxy intercepts the request, checks the validity of the token with the IdM system, and forwards the request to the Key Vault if the token is valid;

9. The Key Vault starts the remote attestation process with the endpoint received, and once the consumer is attested, indicating it runs on a TEE application, it receives the private key (PrK). The Key Vault encrypts the PrK with the symmetric shared key (ShK in Figure 4.5), before sending it to the consumer. This symmetric shared key is exchanged or generated during the remote attestation process;

10. The consumer sends subscription requests to the Publish/Subscribe system to receive notifications (energy consumptions) when the producers generate new data;

11. The PEP Proxy intercepts the request and validates the OAuth2 token with the IdM system. Once the token is valid, the Publish/Subscribe system stores the subscriptions;

12. The consumer receives notifications whenever a producer sends new data (energy consumptions) to the Publish/Subscribe system. As the consumer has the private key (PrK), it can decrypt data and process them in a secure TEE enclave.

Based on the steps defined above, the producers transmit data securely to the Publish/-Subscribe system. An attacker can not discover what sensitive data are transmitted and stored in the Publish/Subscribe system since they are encrypted. The messages exchanged with the Key Vault should also be performed through TLS (HTTPS, for instance).

4.7 Summary

This Chapter presented the trusted architecture proposed, describing the base case scenario and the main security principles considered for this proposal, according to the established threat model. To make this architecture feasible, a TEE application, named Key Vault, manages keys generation, storage, and distribution. Besides, we also suggested a security im-

provement for a PEP Proxy: perform the remote attestation process, guaranteeing that a requester runs on a TEE application, instead of only checking its authorization. In the end, the last Section presented and described the communication flow for producers and consumers.

Chapter 5

Formal Validation and Experimental Evaluation

This Chapter contains the formal validation for the proposed trusted architecture and an experimental evaluation of a proof of concept application. The first Section presents the formal validation, considering the model checking of the Coloured Petri Net model that represents the communication flow exposed in Chapter 4. The model checking verifies the model behavior regarding 14 security properties. Then, the second Section presents the performance evaluation of an IoT application implemented according to the trusted architecture, considering all the components and the corresponding communication between them. We carried out experiments to measure this IoT application performance with the trusted architecture and compared the results with a case in which the same application does not consider any security mechanism. The achieved results present a low time overhead when considering all the security and privacy benefits. This Chapter concludes by showing the achieved results and a short discussion.

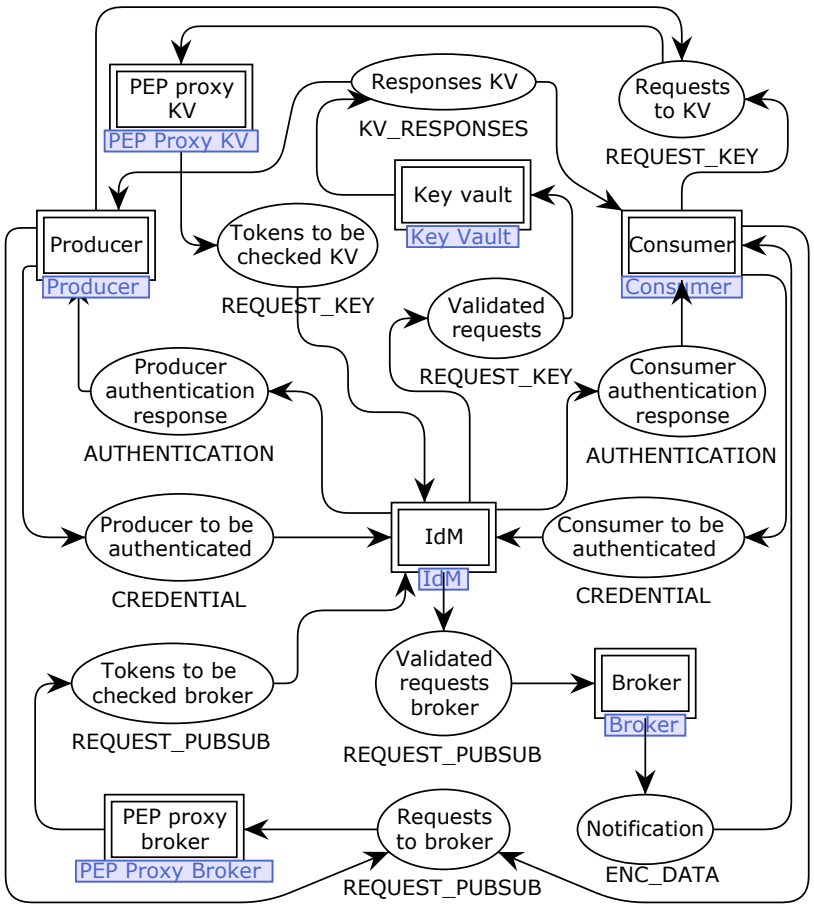
5.1 Formal Validation

This Section presents the Coloured Petri Net (CPN) model [50] of the trusted architecture proposed in Chapter 4, and the application of model checking to verify the security properties. The model represents all the components in the trusted architecture, as well as the communication flow between them.

5.1.1 Trusted Architecture CPN Model

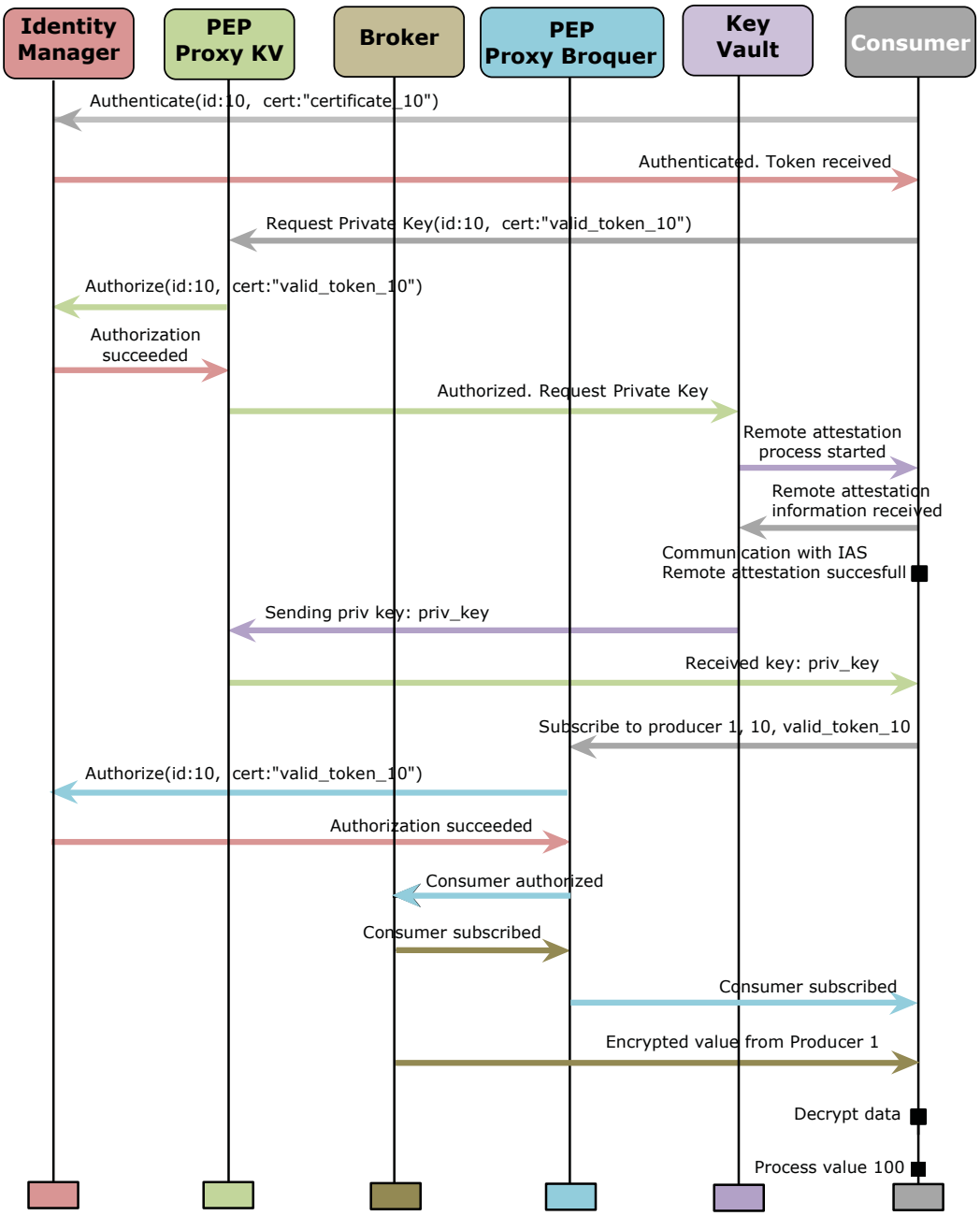
To model the proposed architecture, we considered its main components for the top-level HCPN module, listed as follows: Producer, Consumer, Broker (Publish/Subscribe System), Key Vault, Identity Management (IdM) System, and PEP Proxy. The HCPN model, shown in Figure 5.1, represents these components as substitution transitions. As the trusted architecture requires a PEP Proxy to protect the Key Vault and a PEP Proxy to protect the Broker, the Figure 5.1 presents both as PEP proxy KV and PEP proxy broker, respectively. This way, the top-level HCPN module contains seven substitution transitions, and each of these has a sub-module detailing the specific component.

Figure 5.1: Communication flow model (top-level CPN module)



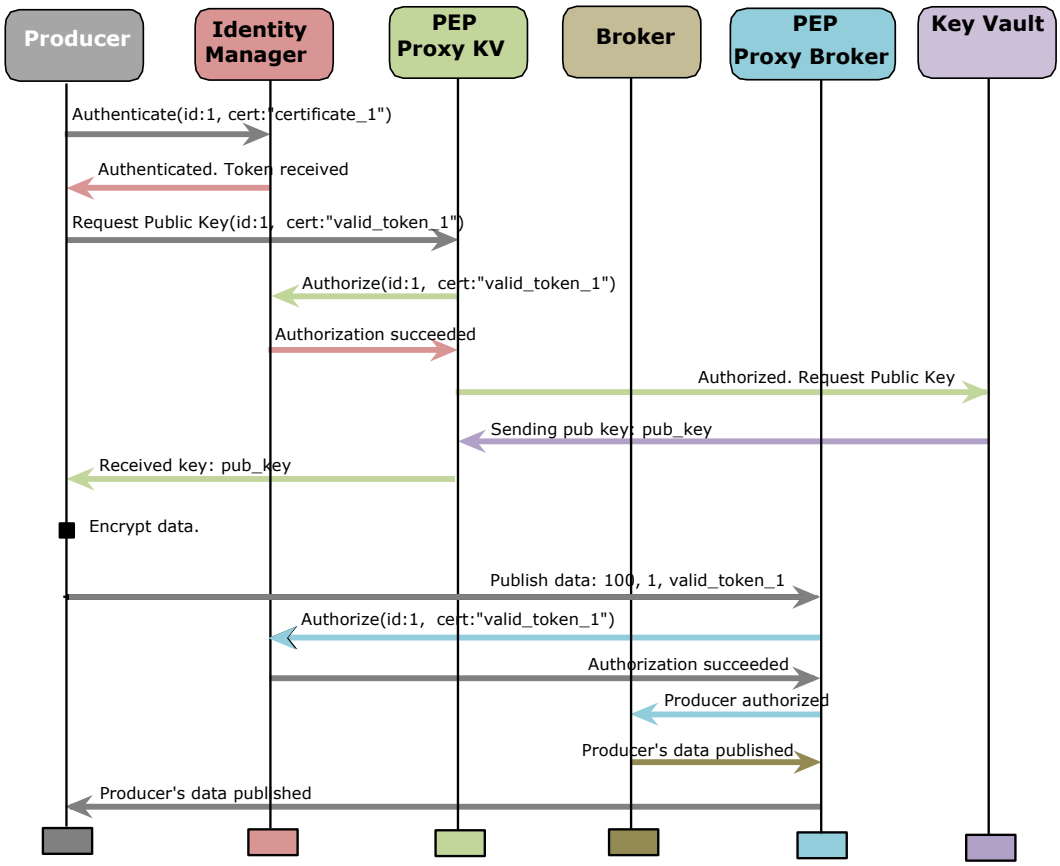
The communication flow was divided into two cycles to ease the understanding and visualization: the producer flow and the consumer flow. Both consider a producer or consumer, and its communications with IdM (authentication), PEP Proxy (authorization), Key Vault (cryp-

Figure 5.2: MSC Consumer flow



tographic keys), and Broker (publish/subscribe system). The Producer and Consumer modules represent each of these communication flows. Figures 5.2 and 5.3 show the message sequence chart/diagram for both flows, consumer and producer respectively, generated when simulating the communications in the model. They contain the events triggered when transitions occur, representing all the communication events between the architectural entities. Figure 5.4 lists the types (colors) defined for this HCPN model.

Figure 5.3: MSC Producer flow



For the consumer flow, the sequence starts with the Consumer being authenticated at the IdM system, when passing valid credentials, such as identification and certificate. This process returns a valid token for the Consumer. With this token, the Consumer requests a private key for the Key Vault, passing by the authorization process at the PEP Proxy KV. Once the PEP Proxy KV authorizes the token, the Key Vault starts the remote attestation protocol, to check that the Consumer runs on a trusted execution environment. Through this process, the Key Vault receives information from the Consumer and verifies with the IAS (Intel Attestation Service, for Intel SGX machines). If the Consumer is a valid SGX, the Key Vault sends the private key. Then, the Consumer subscribes to a Producer in the Broker, also passing by the token authorization process in the PEP Proxy Broker. As soon as the Consumer is subscribed, it receives notifications with encrypted data, whenever the Broker receives updates from the Producer. The Consumer finishes decrypting and processing the data accordingly.

The Producer flow of messages starts with the Producer authentication at the IdM system, a process similar to the Consumer authentication. Using the received token, the Producer

requests a public key for the Key Vault, also passing by the authorization process in the PEP Proxy KV. After the token is validated, the Key Vault sends the public key. The Producer can now encrypt generated data and send them for the Broker, after being authorized at the PEP Proxy Broker too.

Figure 5.4: Types declarations

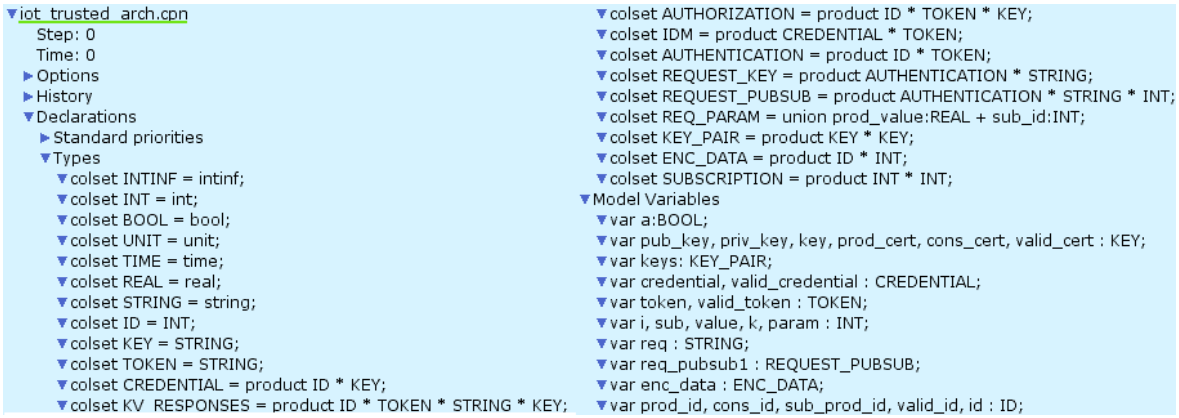
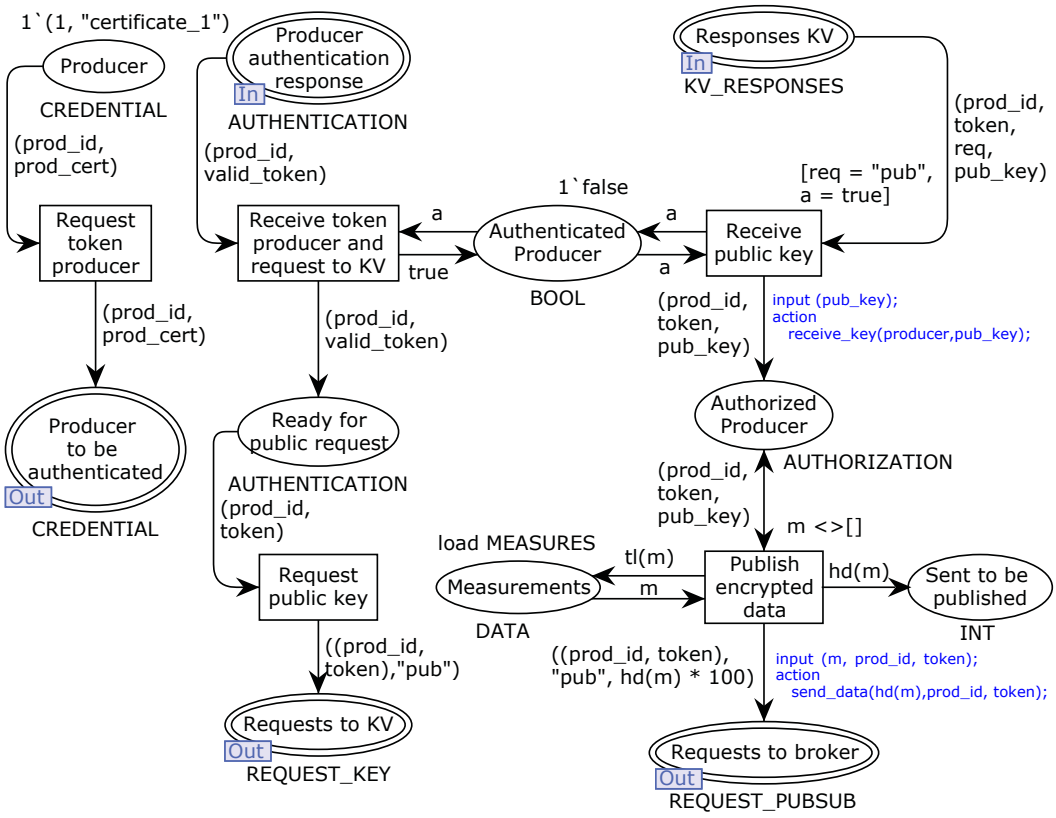


Figure 5.5: Producer flow model



The detailed Producer module, seen in Figure 5.5, contains the following places:

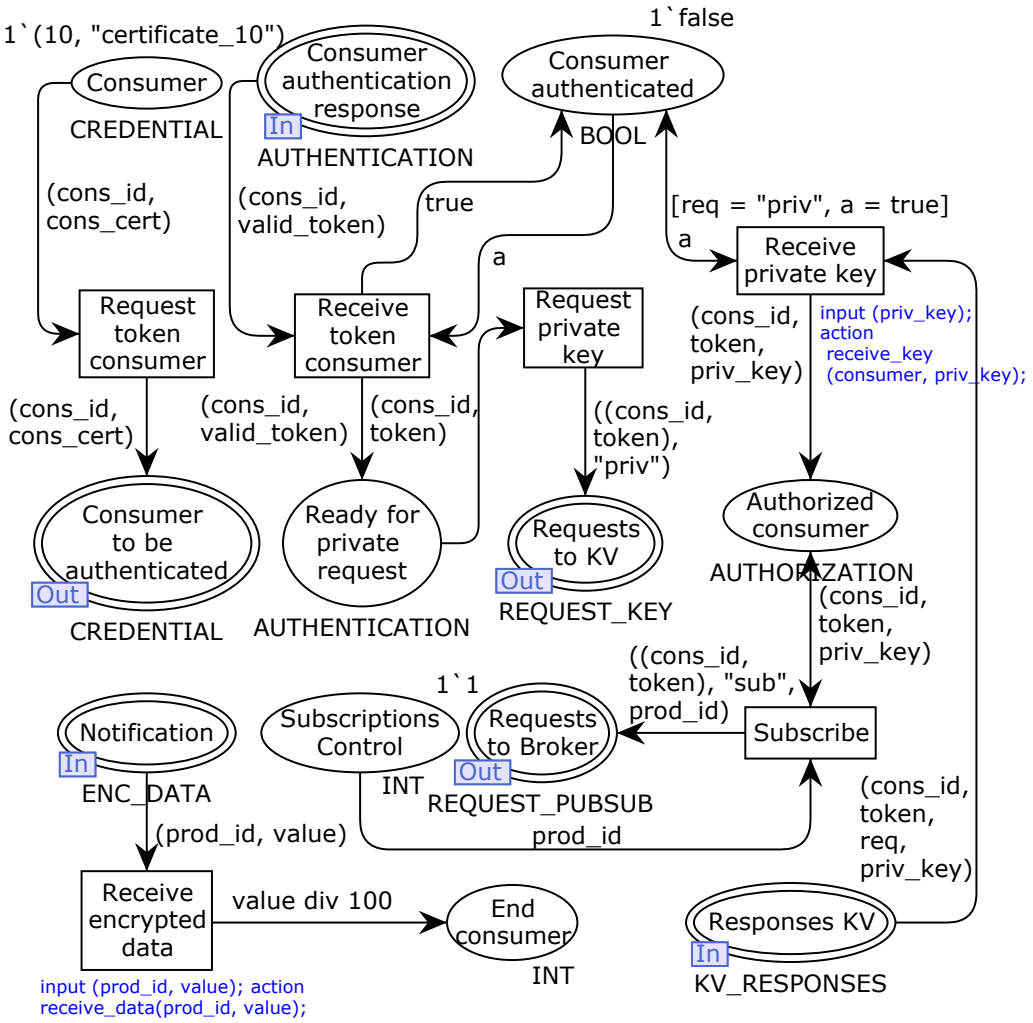
- Producer - the initial state for a producer, containing a token that represents the producer credentials (identification and certificate, for instance);
- Authenticated Producer - an auxiliary place that indicates the Producer was authenticated;
- Ready for public request - this place indicates that the Producer is ready to request the public key, once it was authenticated and received a valid OAuth2 token to access the Key Vault and the Broker;
- Authorized Producer - this place indicates that the PEP Proxy authorized the Producer (OAuth2 token validation) and the Producer received a public key from the Key Vault;
- Measurements - this place represents the measurements generated by producers, i.e., the sensitive data to encrypt before sending to the Broker;
- Sent to be published - this is an auxiliary place to keep the last measurement sent to the Broker.

The following places, shown in Figure 5.5 and shared with the top-level HCPN module (Figure 5.1), are the *port places* (*in* and *out*):

- Producer to be authenticated - an *out* port that passes the Producer credentials to the top-level module, requesting a valid OAuth2 token after the credentials validation;
- Producer authentication response - an *in* port that receives the valid OAuth2 token after the authentication process executed;
- Requests to KV - an *out* port that sends the public key request to be treated by the Key Vault in communication with the top-level module;
- Responses KV - an *in* port that receives the public key after the Key Vault processes the request;
- Requests to Broker - an *out* port that sends encrypted data to be published in the Broker (pub/sub system).

The detailed Consumer module, shown in Figure 5.6, contains the following places:

Figure 5.6: Consumer HCPN module



- Consumer - the initial state for a consumer, containing a token that represents the Consumer credentials (identification and certificate, for instance);
- Consumer authenticated - an auxiliary place to indicate that the Consumer was authenticated;
- Ready for private request - this place indicates that the Consumer is ready to request the private key, once it was authenticated and received a valid OAuth2 token to access the Key Vault and the Broker;
- Authorized Consumer - this place indicates that the PEP Proxy authorized the Consumer

(OAuth2 token validation) and the Consumer received a private key from the Key Vault;

- Subscriptions Control - this is an auxiliary place to control the number of Consumer subscriptions in the Broker;
- End consumer - indicates that the Consumer received, decrypted, and processed the data.

The port places for the Consumer module are described below:

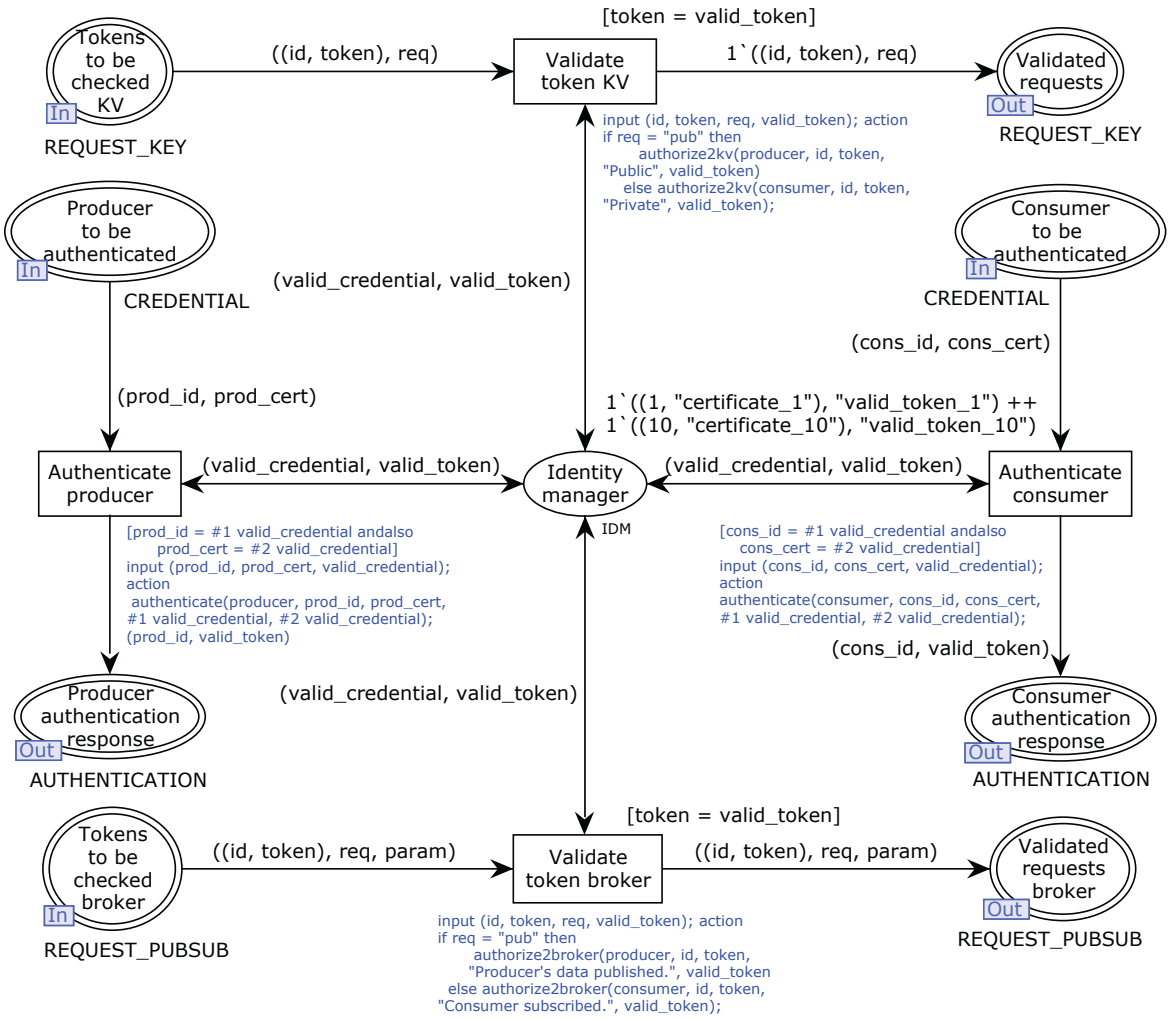
- Consumer to be authenticated - an *out* port that sends the Consumer credentials to be later validated and then get a valid OAuth2 token;
- Consumer authentication response - an *in* port that receives the valid OAuth2 token after the credentials validation;
- Requests to KV - an *out* port that requests the private key to the Key Vault;
- Responses KV - an *in* port that receives the Key Vault response with the private key;
- Requests to Broker - an *out* port that sends the subscription requests to the Broker to allow that the Consumer can receive notifications with data from the Producer;
- Notification - an *in* port that receives the notifications from the Broker, with the Producer's encrypted data.

As shown in Figure 5.1, besides the Producer and Consumer, the model has five other substitution transitions representing the remaining entities for the trusted architecture, namely: IdM (Identity Manager), Broker (Pub/Sub system), Key Vault, PEP Proxy KV (PEP that protects the Key Vault), and PEP Proxy Broker (PEP that protects the Broker).

The IdM HCPN module can be seen in Figure 5.7. Besides the IdM place, which represents the stored information in the IdM system (e.g., Producer and Consumer credentials), this module contains the following port places:

- Producer to be authenticated - this *in* port receives the Producer credentials to be verified in the authentication process;
- Producer authentication response - this *out* port receives the Producer valid OAuth2 token if the authentication process succeeds, i.e., if the Producer credentials are valid;

Figure 5.7: IdM HCPN module



- Consumer to be authenticated - this *in* port receives the Consumer credentials to be verified in the authentication process;
- Consumer authentication response - this *out* port receives the Consumer valid OAuth2 token if the authentication process succeeds, i.e. if the Consumer credentials are valid;
- Tokens to be checked KV - this *in* port receives the OAuth2 tokens, that come in the key requests, intercepted by the PEP Proxy KV to proceed with the authorization process;
- Validated requests - this *out* port sends the tokens successfully validated through the authorization process, indicating that the requests to the Key Vault are authorized;
- Tokens to be checked broker - this *in* port receives the OAuth2 tokens, that come in the

publish/subscribe requests, intercepted by the PEP Proxy Broker to proceed with the authorization process;

- Validated requests broker - this *out* port sends the tokens successfully validated through the authorization process, indicating that the requests to the Broker are authorized.

Figure 5.8: Key Vault HCPN module

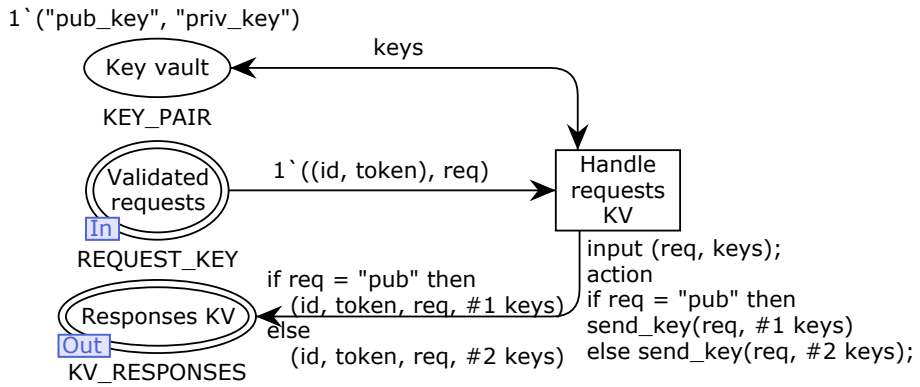


Figure 5.8 shows the Key Vault HCPN module. Besides the Key Vault place, which stores the cryptographic keys, this module contains the following port places:

- Validated requests - this *in* port place receives the authorized requests, after the PEP Proxy KV performs the authorization process;
- Responses KV - this *out* port place sends the keys, public or private, to the PEP Proxy KV according to the requests.

Figure 5.9 shows the HCPN module for the Broker, which has the following places and port places:

- Broker subscriptions - a place that represents the Consumer subscriptions to the Broker, indicating the Consumer's interests in the Producer's data;
- Broker published data - a place to store published data from Producers;
- Validated requests broker - an *in* port that receives authorized requests to publish (Producer) or subscribe (Consumer) in the Broker;

Figure 5.9: Broker HCPN module

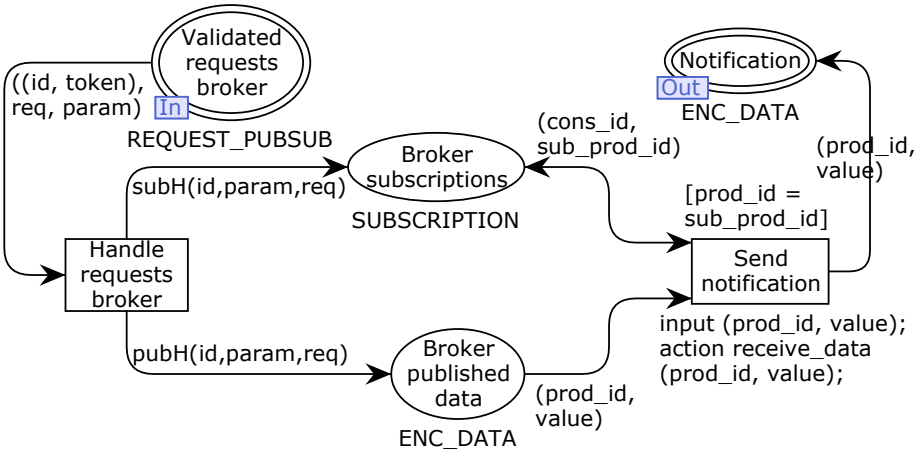
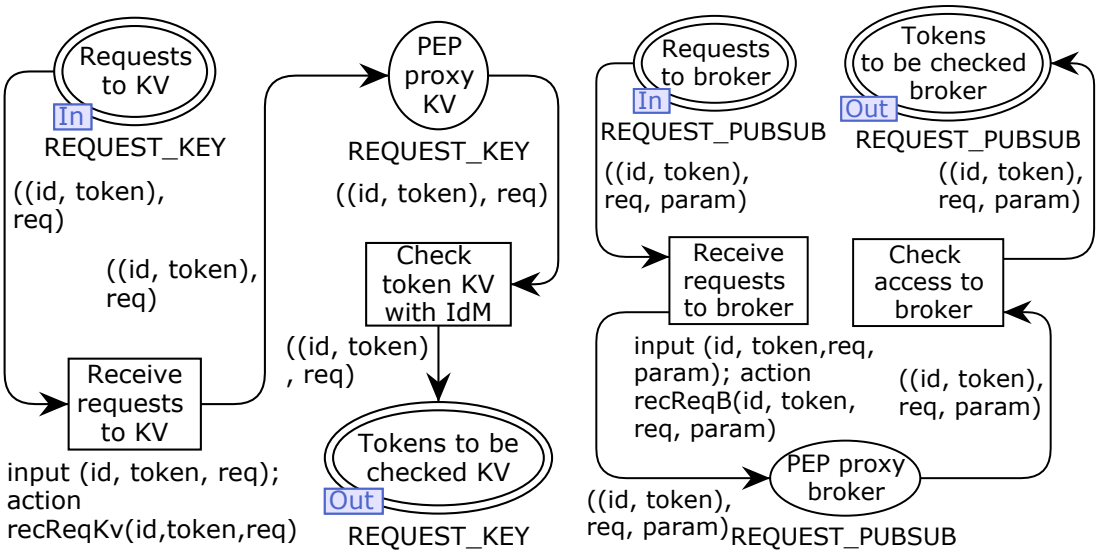


Figure 5.10: PEP Proxy KV and PEP Proxy Broker HCPN modules



- Notification - an *out* port that sends notifications to the Consumer according to the subscriptions.

Figure 5.10 presents both PEP Proxy KV (Key Vault) and PEP Proxy Broker HCPN modules. The places and port places are as follows:

- PEP Proxy KV/broker - this place represents the PEP Proxy, which is responsible for intercepting the requests to the key vault and Broker, and verifying with the IdM if they are authorized;

- Requests to KV/broker - this *in* port receives the requests to the Key Vault and Broker;

- Tokens to be checked KV/broker - this *out* port sends the requests to validate the tokens in the IdM.

5.1.2 Verification - Model Checking

Besides the use of TEEs to protect the sensitive data processing, the security provided by the TIoTA relies mainly on the following processes: authentication and authorization of entities, and cryptography of data. Considering these three processes to perform the model checking, we defined the following properties (the identifiers using ***P_*** are related to the Producer and those using ***C_*** are related to the Consumer):

1. ***P_Authenticated***: guarantees that a Producer with valid credentials reaches an authenticated state, after the authentication process. This indicates that the Producer was authenticated and received a valid OAuth2 token.

2. ***P_Authorized***: guarantees that an authenticated Producer reaches an authorized state, after the authorization process by the PEP/IdM, when requesting a public key. This indicates that the Producer was authorized and already received a public key.

3. ***C_Authenticated***: guarantees that a Consumer with valid credentials reaches an authenticated state, after the authentication process. This indicates that the Consumer was authenticated and received a valid OAuth2 token.

4. ***C_Authorized***: guarantees that an authenticated Consumer reaches an authorized state, after the authorization process by the PEP/IdM, when requesting a private key. This indicates that the Consumer was authorized and already received a private key.

5. ***C_Subscribed***: guarantees that an authorized Consumer reaches a subscribed state, through a subscription in the broker. This marking is represented by the "Broker Subscriptions" place (Figure 5.9), and indicates that an authorized Consumer is interested in receiving Producer's data through Broker notifications.

6. ***Data_Published***: guarantees that an authorized Producer has generated, encrypted, and sent data to the Broker. This marking is represented by the "Broker published data" place

(Figure 5.9), indicating that there are Producer's encrypted data available for sending to the subscribed Consumer, through notification.

7. ***Data_Consumed***: guarantees that an authorized Consumer has received, decrypted, and processed Producer's data. This marking is represented by a token in the End Consumer place (Figure 5.6), meaning that the Consumer successfully received the Producer's encrypted data, through a notification from the Broker, decrypted and processed them accordingly.

Besides the desired behaviors represented by these seven properties, there are behaviors not desired in the model. For instance, according to the considered threat model (Chapter 4, Section 4.2), an unauthorized entity must not succeed to access the protected services (Key Vault and Broker). For this reason, the unauthorized entity cannot publish or subscribe to the Broker and cannot get the right key to decrypt data. Moreover, an entity cannot become authorized if it was not authenticated. To verify that these behaviors do not occur in the modeled communication flow, we defined the following five properties (besides *P_* and *C_*, for Producer and Consumer, the identifiers also use ! and $\wedge$ to represent the logical operations *NOT* and *AND*):

8. ***! (P_Authorized*** $\wedge$ ***! P_Authenticated)*** - guarantees that a Producer cannot be authorized if it was not previously authenticated (i.e., a Producer not authenticated cannot become an authorized Producer).

9. ***! (Data_Published*** $\wedge$ ***! P_Authorized)*** - guarantees that a Producer cannot publish data if it was not previously authorized, and it would not have the public key to encrypt the data (i.e., a Producer not authorized cannot publish data).

10. ***! (C_Authorized*** $\wedge$ ***! C_Authenticated)*** - guarantees that a Consumer cannot be authorized if it was not previously authenticated (i.e., a Consumer not authenticated cannot become an authorized Consumer).

11. ***! (C_Subscribed*** $\wedge$ ***! C_Authorized)*** - guarantees that a Consumer cannot subscribe to a topic in the Broker if it was not previously authorized, and it would not have the private key to decrypt the Producer's data (i.e., a Consumer not authorized cannot subscribe to the Broker).

12. ***! (Data_Consumed ∧ ! C_Subscribed)*** - guarantees that a Consumer does not receive and process data if it was not previously subscribed in the Broker, once only a subscribed Consumer can receive data.

To verify data confidentiality and integrity throughout the communication flow, we defined the two following properties:

13. ***Data_Integrity*** - guarantees that the data processed in the Consumer are the same data generated in the Producer, indicating that the cryptography protects the data during the transmission process, avoiding unauthorized changes to the data (tamper-resistance).

14. ***Data_Confidentiality*** - assures that no unauthorized entity can read or explore the data during the transmission process from the Producer to the Consumer, indicating the protection by the data encryption, i.e., the data are kept confidential during transmission.

Table 5.1: CTL Formulas

Property	CTL Formula
1) ***P_Authenticated***	$EF(producer\_authenticated)$
2) ***P_Authorized***	$EF(producer\_authorized \wedge producer\_authenticated)$
3) ***C_Authenticated***	$EF(consumer\_authenticated)$
4) ***C_Authorized***	$EF(consumer\_authorized \wedge consumer\_authenticated)$
5) ***C_Subscribed***	$EF(consumer\_authorized \wedge consumer\_subscribed)$
6) ***Data_Published***	$EF(data\_sent\_to\_broker \wedge data\_published)$
7) ***Data_Consumed***	$\neg EF(data\_published \wedge consumer\_subscribed \wedge \neg data\_consumed)$
8) ***! (P_Authorized ∧ ! P_Authenticated)***	$\neg EF(producer\_authorized \wedge \neg producer\_authenticated)$
9) ***! (Data_Published ∧ ! P_Authorized)***	$\neg EF\ (data\_published \wedge \neg producer\_authorized)$
10) ***! (C_Authorized ∧ ! C_Authenticated)***	$\neg EF(consumer\_authorized \wedge \neg consumer\_authenticated)$
11) ***! (C_Subscribed ∧ ! C_Authorized)***	$\neg EF(consumer\_subscribed \wedge \neg consumer\_authorized)$
12) ***! (Data_Consumed ∧ ! C_Subscribed)***	$\neg EF(processed\_data \wedge \neg consumer\_subscribed)$
13) ***Data_Integrity***	$\neg EF(tampered\_data \wedge processed\_data\_not\_empty)$
14) ***Data_Confidentiality***	$\neg E(decrypted\_value\ U\ processed\_by\_consumer)$; $\neg EF(decrypted\_at\_broker)$; $\neg EF(decrypted\_at\_pep)$

To perform the model checking, we first described the properties in CTL, and then we migrated the represented formulas to ASK-CTL, which has a specific integration in the

CPN Tools. Table 5.1 presents the representation of the defined properties in CTL. For the first property, ***P_Authenticated***, we defined the CTL formula *EF (producer_authenticated)*, which means that, in some future moment (*EF*), we will get a representation for an authenticated Producer (*producer_authenticated*). When translating this CTL formula to ASK-CTL representation, we map this (*producer_authenticated*) to a specific marking in a specific place (Authenticated Producer) of the Producer HCPN module (Figure 5.5). For the other properties, we followed the same logic to represent them in CTL formulas.

The properties related to data integrity and confidentiality assures that data cannot be tampered or read during the communication flow, once the Producer encrypted them previously. The properties also assure that only an authorized Consumer can decrypt the data. This way, if the system suffers a "man-in-the-middle" attack, the information will not be obtained since the attacker does not have the right key to decrypt the data.

The other properties are related to desired and undesired behaviors (considering the TIoTA threat model - Chapter 4, Section 4.2), representing, for instance, a possible attacker that wants to publish data or subscribe to the Broker but does not have valid credentials. As a consequence, once the authentication process will not succeed, the attacker will not be authorized to communicate with protected services.

Through the model checking process, all the 14 defined properties were satisfied, giving evidence that the trusted architecture keeps the data secure. The properties related to the authentication and authorization processes guarantee that an entity should have valid credentials to access protected services, such as the Key Vault and the Broker. Thus, only authenticated Producers will be authorized to receive public keys from the Key Vault and to publish data in the Broker. Also, only authenticated Consumers will be authorized to receive private keys from the Key Vault and to subscribe to the Broker.

With the specification verified and validated, we can apply model-based testing (MBT) to generate test cases, providing project artifacts to improve confidence in the correct implementation of the TIoTA. These test cases can assist developers when evaluating the conformance of an implementation with the TIoTA formal specification. To exemplify, the generated tests can comprise the following scenarios: (i) with correct credentials, the producer authenticates successfully; (ii) once the producer publishes the data, the data are kept encrypted during transmission; and, (iii) after the producer transmits the data, the same data

are received by the consumer. The MBT application is out of scope for this work.

5.2 Experimental Evaluation - Proof of Concept

This Section presents an implemented IoT application that considers all the components of the proposed trusted architecture, the adopted scenario (IoT devices as Producers, Broker, and Consumer), and the set of experiments considered to assess the performance of such application. To instantiate the architecture, we applied three FIWARE components (Orion Context Broker, Keyrock Identity Management, and Wilma PEP Proxy), as described in the next Subsection.

5.2.1 FIWARE

FIWARE (Future Internet-WARE) is a software platform created to ease the development of smart applications for the Future Internet in multiple sectors and following open standards [28]. For this, it provides a wide set of APIs (Application Programming Interfaces). FIWARE has many software components, called Generic Enablers (GEs), with some of them being based in the well-established cloud platform OpenStack. OpenStack is also used to establish a cloud that provides FIWARE services.

For each FIWARE component (GE) specification, there is an open-source reference implementation. The FIWARE GEs cover the following domains [32]: Data and Context Management, Internet of Things Services Enablement, Security, Cloud Hosting, and Advanced Web-based User Interface. The FIWARE GEs of interest for this work are Orion Context Broker (CB), Keyrock Identity Manager, and Wilma PEP (Policy Enforcement Point) Proxy, once we used them for the proof of concept implementation of this thesis.

Orion CB is a context data management system, allowing the operations of registering, updating and querying context data [29]. It works with the publish/subscribe communication pattern, allowing the sending of notifications to interested parties when changes occur in the entities of interest. The Keyrock Identity Manager is an identity and access management (IAM) system, being responsible for registering and querying identity data. It provides access tokens for users, after the authentication process, and validates them to allow or deny access to protected services [31]. Wilma PEP Proxy is a Policy Enforcement Point responsible

for simple authorization, controlling the access to protected services by validating received tokens with the Keyrock Identity Manager [30].

5.2.2 IoT Application

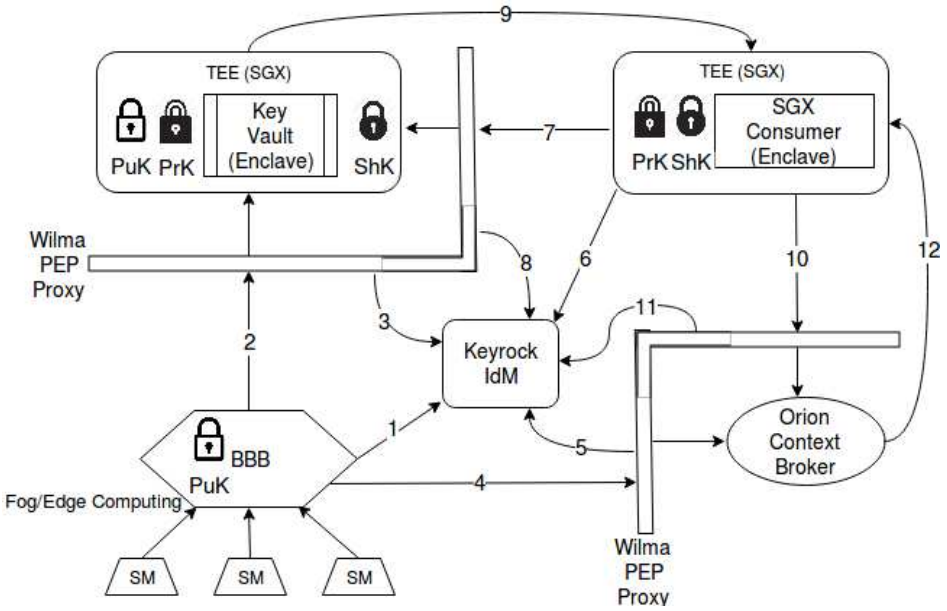
The implemented IoT application uses the following technologies:

- FIWARE Components:
  - Orion Context Broker, as the Broker (in this case, Publish/Subscribe System);
  - Keyrock, as the Identity Management System;
  - Wilma, as the PEP Proxy;
- Intel SGX applications:
  - Consumer;
  - Key Vault.

For this proof of concept application, the Orion Context Broker acts as a Publish/Subscribe Broker, receiving the energy consumption data from Producers (smart meters) and sending notifications to Consumers (aggregators), while the Keyrock and Wilma perform the authentication and authorization processes, protecting the Orion Context Broker and the key manager (Key Vault).

Figure 5.11 shows all the instantiated components for this IoT application with respective communication flow, which was detailed in the Chapter 4, Section 4.6. In this proof of concept, to validate that the trusted components (Consumer and Key Vault) run inside Intel SGX enclaves, we perform the remote attestation (RA) process that checks with the Intel Attestation Service if these enclaves are valid and if they are running on a real Intel SGX. This process is executed by Producers, attesting the Key Vault whenever it is necessary to get a public key, and by Key Vault, attesting the Consumer whenever it receives a private key request.

Figure 5.11: Application Communication Flow with instantiated components.



5.2.3 Experiments Setup

To evaluate this IoT application performance, we deployed all the components of the proposed architecture in two machines equipped with Intel Skylake CPU model i7-6700, 3.4GHz, 8MB cache, and 8GB RAM (Dell OptiPlex 5040). To have a more realistic scenario, we distributed all the components as follows: one machine with the Producers and the other with the Orion Context Broker deployed at a virtual machine, Consumer and Key Vault at separated containers, as well as KeyRock IdM and Wilma PEP Proxy.

After this setup, we ran a set of experiments involving the communication flow explained previously: Producers generated and sent data to the Orion Context Broker, and the Consumer received these data through notifications from the Orion Context Broker.

5.2.4 Scenarios, Metrics, and Parameters

To analyze the performance of the implemented solution, we considered the following two scenarios:

- Without data security, considering just Producers, Consumer and Orion Context Broker (without any security mechanism);

- With data security, including the Key Vault for keys management and distribution, encryption/decryption enabled by these keys, and the authentication/authorization enabled by the Keyrock and the Wilma PEP Proxy.

These scenarios were considered to compare the trusted architecture with a case using the regular architecture, without employing any security mechanisms. The idea was to verify the performance overhead that is increased by the protection of our proposal. Besides, we suggest using these scenarios whenever necessary to compare distinct security solutions, once each solution provides different operations and technologies and generally runs in different environments. This way, regardless of the security solution, we will always normalize the results comparing the performance overhead between an application that employs the security operations and an application that employs the basic architecture without any security operation. By applying the same basic architecture for comparison with the different security solutions, we can now have a fair comparison among them since the overhead is measured regarding the execution of the application with the basic architecture.

As every security operation increases the performance overhead, increasing the processing time, we decided to measure the latency for both scenarios. The experiments obtained this metric as follows: the elapsed time for data leaving the Producer and reaching the Consumer; as seen before, to accomplish this, the data must leave the Producer and arrive in the Consumer through notifications from the Orion Context Broker, in the first scenario; the experiments repeat this flow for the second scenario, considering the keys distribution and the encryption/decryption/authentication/authorization processes.

For both scenarios, we executed 500 publication cycles to obtain a reliable sample since the results did not present significant variability after 100 publication cycles (we checked with 100, 200, 300, 400, and 500 publication cycles). For each execution, we marked the time when the Producer generated the data, and the time when the Consumer processed the data. Besides the communication flow latency for both scenarios, we also measured the remote attestation latency, since the remote attestation is an important process performed in the communication with the Intel SGX applications: Key Vault and Consumer.

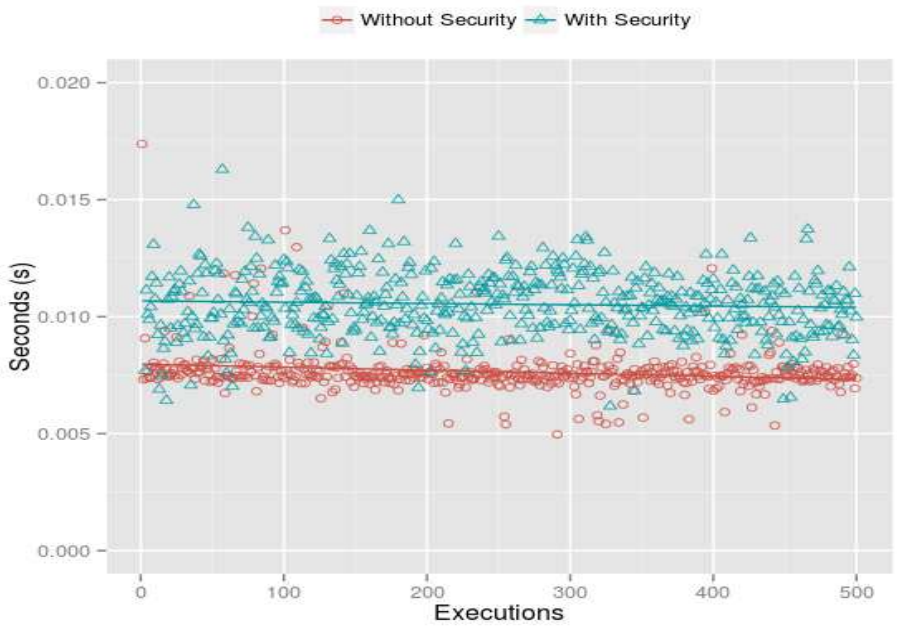
Finally, we also carried out tests related to the scalability of our solution, concerning the number of possible Producers sending measurements concurrently. We executed tests with 100, 200, 300, and 400 producers because with more than 400 Producers the system became

overloaded, making the communication almost impractical. Every test cycle was executed by ten minutes, registering the latency values for the concurrent communications.

5.2.5 Results and Discussion

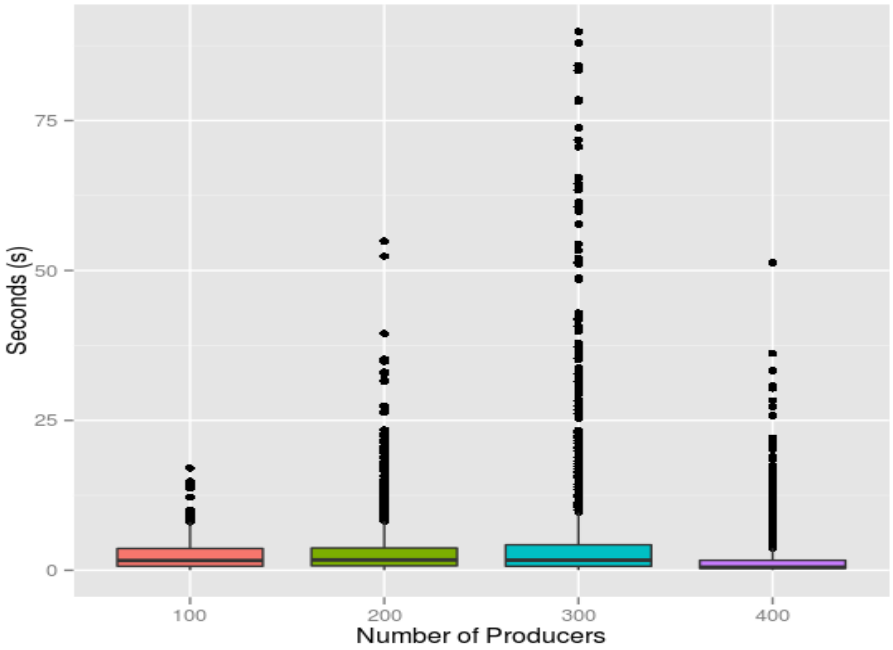
According to the achieved results, as shown in Figure 5.12, our proposed solution for data security presents low overhead related to a solution that does not use any security mechanism: approximately 10.5 milliseconds against 7.5 milliseconds on average. The remote attestation process obtained an average time of 2.5 seconds, but this process is executed only once to get the keys (once Producer and Consumer have the respective keys, they do not need to communicate with Key Vault).

Figure 5.12: Latency



This solution also presents a good level of scalability since it succeeded to keep working even with different loads of requests from different numbers of producers at the same time: 100, 200, 300, and 400. However, as seen in Figure 5.13, the higher the number of producers at the same time, the higher the latency to process the requests and the lower the number of processed requests. This result is dependent on our testbed infrastructure, considering its setup. Obviously, if we use more robust machines and components, we will get better results.

Figure 5.13: Latency considering many producers at same time (stress)



Considering the threat model applied and experiments, we can say that our solution provides the following features:

- Integrity – the produced data are preserved during all the communication flow, i.e., they cannot be modified without being detected;
- Privacy – the privacy of the customers can be preserved since the produced data cannot be read in any of the communication flow steps, even if the Orion CB is under the control of an attacker;
- Confidentiality – the confidentiality of the data is preserved because they can not be read/accessed, except by authenticated and authorized producers and consumers;
- Authentication – all producers and consumers must use their credentials to get access to the protected services and thus successfully perform the communication flow;
- Secure communication – data in transit and channel are encrypted.

These properties are achieved thanks to cryptography techniques, TLS/HTTPS protocol, and Intel SGX technology. Assuming that the producer's credentials are safe/protected, our

solution also satisfies the non-repudiation property in a way any producer can not deny any previous sending of data.

5.3 Possible Limitations

This Section presents possible limitations of this thesis' validation. In the following subsections, we describe the threats to the validity of the formal and experimental validations, mentioning how we dealt with them.

5.3.1 Threats to the Formal Validation

Every model is always a limited representation of a real scenario or a system due to the considered simplifications. Indeed, a model is good as it allows us to analyze and verify a system's desired behaviors as defined by a set of specifications. Thus, our HCPN model is also limited as it simplifies, for instance, the internal behaviors of the Key Vault, IdM, Broker, and PEP Proxy. However, our model represents all the trusted architecture's components and every relevant communication among them, and the security processes (authentication, authorization, and cryptography). This representation is enough to observe how the processes occur and how the architecture provides data security.

The model checking process relies on the definition of properties representing the model's desired and undesired behaviors. Thus, if an essential property is not defined, the model checking will not achieve suitable coverage, i.e., the process will not check all the model's relevant behaviors. Although we could define more properties, maybe regarding any alternative communication flow, we are sure the 14 presented properties cover all the trusted architecture's security processes since we used the considered threat model (Chapter 4, Section 4.2) as a basis for the properties specification. Thereby, we consider those properties sufficient to demonstrate how the proposed architecture increases data security. Of course, it may be the case that we did not cover all the systems' states, and there is a possibility of having some undesired behavior that we tried to cover by defining not only desired behaviors but also undesirable ones. Also, even when the system is proved to be correct, there is still a question of how complete the specification is and whether it covers all the system's behaviors. A more in-depth discussion on the coverage of model checking coverage is beyond the

scope of this thesis, and the reader interested in the problem can refer to Chockler et al. [22; 23; 24].

5.3.2 Threats to the Experimental Validation

The experimental scenario is not a real IoT application with many devices acting as data producers. This way, we decided to distribute the application's components in two machines. Besides, the experimental setup also considered components executing inside containers and virtual machines. This setup makes the scenario more realistic than if we had deployed everything in just one computer. Although the IoT application did not use real devices, we can estimate the overhead added by our trusted architecture, respecting the existent proportions. This is feasible because we normalized the experiments by comparing the trusted architecture's results with the basic architecture's results. Therefore, if we run experiments with two real IoT applications, one using the trusted architecture and another using the basic architecture, we expect a similar proportion limited by the devices' processing power. This limitation will impact mainly the data encryption performed in the first part of the communication flow (the other security processes should not be impacted as they do not use to run inside devices).

To measure the latency, we started running 100 communication cycles and calculating the average latency. Then, we increased the number of executions observing the variation in the calculated mean. After running 100, 200, 300, and 400 cycles measuring the communication latency, we noticed that mean latency did not suffer significant variation. Then, we concluded our experiment with a reliable sample after executing 500 communication cycles. To test the application's scalability, we followed a similar approach: we started measuring the latency values during the communication of 100 producers sending measurements concurrently to the broker. Then we increased the number of concurrent producers by 100 until the system became overload, not processing many of the sent requests. This way, we executed every experiment cycle for ten minutes, considering 100, 200, 300, and 400 producers (the system became overloaded after 400).

Lastly, regarding the use of TEEs, we are aware that the technologies available in the market also have vulnerabilities and are not a bullet-proof solution. As stated previously, TEE is vulnerable to side-channel attacks (Chapter 2, Section 2.3 and Chapter 3, Subsection 3.2.7). Nevertheless, even considering these vulnerabilities, we ensure our architecture pro-

vides what it should: means to protect data in cloud/fog-based IoT applications according to the defined research questions and proposed problem (Chapter 1, Section 1.2). We wanted to limit the users' distrust in server providers, increasing data security, and a TEE helps in this sense since it makes the attacker's work harder.

5.4 Summary

This Chapter described the validation of the proposed trusted architecture and presented the achieved results. Firstly, it described the HCPN model, which considered all the trusted architecture entities and their main interactions. Then, it presented the verification and analysis of 14 security properties through model checking. It also described an implemented application that considers the proposed architecture. Experiments measured its performance, considering the latency of the communication flow and its scalability. The results presented a low overhead compared to the same application considering an architecture without any security mechanism. The last Section presented the possible limitations of the formal and experimental validations.

Chapter 6

Conclusion

As highlighted previously (Chapter 1), data security in the IoT field is still a challenge. In this thesis, we considered a scenario based on a smart metering system that generates sensitive data. Due to NIALM techniques, this system can be a target of privacy attacks since an attacker can infer behavior from a house by just analyzing its energy consumption measurements. The principal motivation for this work is the lack of a well-established architecture with security methods and components to protect data in cloud/fog-based IoT applications.

Trusted Execution Environments (TEEs) have been applied to improve data security in different application scenarios, considering cloud, fog, and edge computing. To analyze how this technology is employed to protect data in IoT applications, we carried out a systematic literature review (Chapter 3, Section 3.2). Thus, we defined a protocol to ease the replication of this study, and, according to it, we selected 58 works from important scientific repositories, considering journal and conference papers.

We presented the overall results related to the collected data, including quantitative information (Appendix A, Section A.2), and the research results related to the research questions (Chapter 3, Section 3.2.2). Answering each of the four defined questions (Chapter 3, Section 3.2.1), based on the papers' information, we discussed the IoT solutions and application scenarios of TEEs. We then identified the advantages and disadvantages of the use of TEEs and the suggested future works. We grouped the IoT solutions into three main types: OS, Application, and Security. The OS solutions propose protection related to components directly associated with the OSes; the Application solutions propose data protection in typical applications; and the Security solutions propose protection to security mechanisms, such as

authentication and attestation. Besides, we discussed some challenges and directions for new researches on the adoption of TEEs (Chapter 3, Section 3.2.7).

In this context, we decided to provide an architecture considering trusted computing (TEEs) and basic security aspects, such as authentication, access control, and cryptography. We proposed a trusted architecture considering the base case scenario (Chapter 4, Section 4.1), in which producers (IoT devices) generate and send data to a broker (e.g., a publish/-subscriber system), and consumers subscribe to this broker to receive notifications with the producers' data. The proposed trusted architecture (Chapter 4, Section 4.6) requires:

- authentication and authorization for both producers and consumers;
- a TEE application for consumers;
- a TEE application for cryptographic keys generation, storage and distribution;
- secure data communication.

To generate, store, and distribute cryptographic keys, we proposed the Key Vault (Chapter 4, Section 4.4). Key Vault is secure and trusted because it runs on a TEE (e.g., Intel SGX). In our architecture, the Key Vault substitutes the use of a Public Key Infrastructure (PKI) and, thanks to its keys distribution and the consequent cryptography used, we achieve data integrity and confidentiality, enabling the development of privacy-friendly solutions.

Access control mechanisms are present in many systems to allow access only to authorized parties/entities. A PEP Proxy can be used to intercept requests and verify if the requesters have permission to access the protected data/service by checking tokens, policies, or other attributes. In general, a PEP Proxy only validates if the requester is authenticated and has the authorization to access what is requested. In this thesis, we also proposed an improvement to PEP Proxies (Chapter 4, Section 4.5): adding the implementation of the remote attestation protocol to verify if the requester runs in a TEE, guaranteeing that data will be processed more securely, which is warranted by the TEE capabilities and protections. This way, only authenticated and authorized requesters that run in a TEE have access to the protected data/services.

To formally verify important security aspects regarding the architecture operation, we used a Hierarchical Coloured Petri Net to model its components and the overall communi-

cation flow (Chapter 5, Section 5.1.1). For this, we considered the communication among all the architecture components and the authentication, authorization, and encryption operations. Then, according to the threat model (Chapter 4, Section 4.2), we defined 14 security properties related to desired and undesired behaviors of the architecture and performed the CTL model checking to verify the satisfaction of these properties (Chapter 5, Section 5.1.2). Once all the properties were satisfied through the model checking process, we have evidence that the trusted architecture assures data confidentiality and integrity, preserving data security and privacy.

To experimentally evaluate the overhead of our proposal, we implemented an application considering the trusted architecture and measured its performance (Chapter 5, Section 5.2). This implemented version used three FIWARE components (Orion, Keyrock, and Wilma) and Intel SGX. Through experiments, we measured the latency during all the communication flow between a producer and a consumer and measured the solution scalability when dealing with many producers generating and sending data concurrently.

The achieved results show that our solution has a low overhead when compared to an architecture that does not consider any data security mechanism. Using all the employed security components (Key Vault, IdM, PEP Proxy, and their related processes), our solution had an increase of just three milliseconds in the latency of the messages flow, i.e., the elapsed time between producers and consumers considering all the communication flow with the operations of getting keys, encrypting, sending, and decrypting data. Moreover, our solution does not impose scalability constraints additional to the underlying infrastructure. The Key Vault component and the needed attestation tasks can be easily made scalable due to its simplicity. Besides, our stress test reached 400 producers sending data almost at the same time. We also measured the time spent with the remote attestation (RA) process that validates if a system runs on a valid Intel SGX. The RA process presented a mean time of 2.5 seconds, what is considered acceptable for our application, once it is commonly performed just one time for each new producer or consumer, to validate the Key Vault or the new member, respectively.

With these results, we answered the two research questions (Chapter 1, Section 1.2) since we proposed a trusted architecture for IoT applications by specifying what security components are necessary and how every component of the architecture should communicate with

others (RQ1), and we proposed the use of TEEs to improve the security of data processing, decreasing the users' distrust in cloud/fog servers (RQ2). Besides, we achieved the main objective, which was providing an architecture to protect data in IoT applications (Chapter 1, Section 1.3). For this, we achieved each of the specific objectives: applying authentication and authorization components; applying cryptography to protect data; proposing a component for keys management; proposing the TEE usage to process data securely; specifying the communication flow for the components; creating a CPN model that represents the architecture components and communication flow; performing model checking to verify security properties; and implementing a proof of concept application to verify its performance.

These results were published at IEEE Symposium on Computers and Communications (ISCC) 2018, with the paper entitled "Achieving Data Dissemination with Security using FIWARE and Intel Software Guard Extensions (SGX)", which was awarded as the "Best Local Paper". We also presented these results as a poster in the LANCOMM (Latin American Student Workshop on Data Communication Networks) 2019, which was awarded as one of the six best posters. We recently submitted three other papers to different journals. The first paper is a survey regarding the use of TEE for IoT applications. The second paper is the SLR. And the last paper presents the formal validation of the trusted architecture (HCPN model and the model checking of security properties).

6.1 Future work

In our experimental evaluation, we used two FIWARE Security GEs: KeyRock Identity Management and Wilma PEP Proxy. Both can be considered untrusted since they run unprotected in the cloud and can be explored by adversaries. As future work, we envisage that these GEs can be implemented with some TEE technology. Furthermore, we can add Attribute-Based Encryption on the Key Vault to control access to the keys using intrinsic attributes of the IoT devices. In this case, the access to keys would be according to the entities' attributes.

As future work for our systematic literature review, we envisage its extension considering the snowballing technique to add even more papers. Besides, as we recently noticed a growing number of publications, we can update the results with recent researches. Another possibility is replicating this study for other application scenarios since we focused on

cloud/fog-based IoT scenarios.

Regarding the access control, we suggest that a PEP Proxy be implemented in a TEE system, ensuring that the requests are processed with more security. Besides, a timestamp with an expiration time can be added to attested requesters to identify when a new attestation process is required.

For the remote attestation protocol, we suggest its modeling and model checking also considering security aspects. Regarding our HCPN model, we can extend it to include, for instance, time information based on experiments, allowing the verification regarding scalability and performance properties. The model checking process also enables the creation of test cases to verify if a given implementation satisfies the security properties, i.e., if the implementation is compliant with the architecture.

Lastly, we also suggest implementing the trusted architecture using another TEE technology: ARM TrustZone. As ARM TrustZone runs in many IoT devices, it is more proper for IoT applications than Intel SGX. Moreover, we suggest using other software components as IdM, PEP Proxy, and publish/subscribe system. Our previous investigation has led us to consider the OP-TEE as the TrustZone technology once it is open source and has a relatively active community. We also performed some previous tests regarding the environment configuration as well as the implementation and execution of simple applications.[1] [2] [3] [4]

[1] Video showing the environment running samples: `https://youtu.be/pKl_R1DsgAg`

[2] Video showing an RSA example running: `https://www.youtube.com/watch?v=XK6uWNi7VNs`

[3] RSA example in the Github: `https://github.com/cezane/optee_rsa_example`

[4] OPTEE build tutorial: `https://www.slideshare.net/daltoncezane/optee-on-qemu-build-tutorial`

Bibliography

[1] Tigist Abera, N. Asokan, Lucas Davi, Jan-Erik Ekberg, Thomas Nyman, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. C-flat: Control-flow attestation for embedded systems software. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, page 743–754, Vienna, Austria, 2016. ACM.

[2] N. Ahmed, M. A. Talib, and Q. Nasir. Program-flow attestation of iot systems software. In *15th Learning and Technology Conference (LTC)*, pages 67–73, Feb 2018.

[3] N. Akhtar, A. Rehman, M. Hussnain, S. Rohail, M. S. Missen, M. Nasir, A. Hayder, N. Salamat, and M. Pasha. Hierarchical coloured petri-net based multi-agent system for flood monitoring, prediction, and rescue (fmpr). *IEEE Access*, 7:180544–180557, 2019.

[4] F. Alsubaei, A. Abuhussein, and S. Shiva. Quantifying security and privacy in internet of things solutions. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, April 2018.

[5] Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar. Physical Unclonable Functions for IoT Security. In *IoTPTS@AsiaCCS*, pages 10–13. ACM, 2016.

[6] Yisheng An, Naiqi Wu, Xiangmo Zhao, Xuan Li, and Pei Chen. Hierarchical colored petri nets for modeling and analysis of transit signal priority control systems. *Applied Sciences*, 8(1), 2018.

[7] R. Ankele and A. Simpson. On the performance of a trustworthy remote entity in comparison to secure multi-party computation. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 1115–1122, 2017.

[8] Arm. TruztZone - Arm Developer. Available: `https://developer.arm.com/technologies/trustzone`. Accessed: 2019-01-13.

[9] Kevin Ashton. That 'Internet of Things' Thing. Available: https://www.rfidjournal.com/articles/view?4986, 2009. Accessed: 2019-01-07.

[10] Hasiba Ben Attia. *Formal Modelling and Verification of Security Policies in Cloud Computing*. PhD thesis, University Mohamed Khiderof Biskra, Biskra, Algeria, 2019.

[11] Pierre-Louis Aublin, Florian Kelbert, Dan O'Keeffe, Divya Muthukumaran, Christian Priebe, Joshua Lind, Robert Krahn, Christof Fetzer, David M. Eyers, and Peter R. Pietzuch. TaLoS : Secure and Transparent TLS Termination inside SGX Enclaves. In *Report number: 2017/5. Affiliation: Imperial College London*, 2017.

[12] G. Ayoade, V. Karande, L. Khan, and K. Hamlen. Decentralized iot data management using blockchain and trusted execution environment. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 15–22, July 2018.

[13] Pedro Yóssis Silva Barbosa. *Privacy by Evidence: A Software Development Methodology to Provide Privacy Assurance*. PhD thesis, Universidade Federal de Campina Grande, 2018.

[14] Adom Beck. Python SGX. Available: `https://github.com/adombeck/python-sgx`. Accessed: 2020-01-13.

[15] Yosra Ben Mustapha, Hervé Debar, and Gregory Blanc. Policy Enforcement Point Model. In *SECURECOMM 2014: 10th International Conference on Security and Privacy in Communication Networks*, pages 278 – 286, Beijing, China, September 2014. Springer International Publishing.

[16] Elisa Bertino. Data Security and Privacy: Concepts, Approaches, and Research Directions. In *40th IEEE Annual Computer Software and Applications Conference, COMPSAC 2016, Atlanta, GA, USA, June 10-14, 2016*, pages 400–407, 2016.

[17] Chen Cao, Le Guan, Ning Zhang, Neng Gao, Jingqiang Lin, Bo Luo, Peng Liu, Ji Xiang, and Wenjing Lou. *CryptMe: Data Leakage Prevention for Unmodified*

Programs on ARM Devices: 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings, pages 380–400. 09 2018.

[18] Z. Berkay Celik, Patrick McDaniel, and Gang Tan. Soteria: Automated iot safety and security analysis. In *USENIX Annual Technical Conference*, Boston, MA, 2018. USENIX Association.

[19] R. Chang, L. Jiang, W. Chen, Y. Xie, and Z. Lu. A trustenclave-based architecture for ensuring run-time security in embedded terminals. *Tsinghua Science and Technology*, 22(5):447–457, Sep. 2017.

[20] R. Chang, L. Jiang, Q. Yin, W. Liu, and S. Zhang. A novel method of apk-based automated execution and traversal with a trusted execution environment. In *International Conference on Computational Intelligence and Security (CIS)*, pages 254–258, 2016.

[21] Rui Chang, Liehui Jiang, Wenzhi Chen, Yang Xiang, Yuxia Cheng, and Abdulhameed Alelaiwi. MIPE: a practical memory integrity protection method in a trusted execution environment. *Cluster Computing*, 20:1075–1087, 2017.

[22] Hana Chockler, Orna Kupferman, Robert P. Kurshan, and Moshe Y. Vardi. A practical approach to coverage in model checking. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification*, pages 66–78, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[23] Hana Chockler, Orna Kupferman, and Moshe Y. Vardi. Coverage metrics for formal verification. In Daniel Geist and Enrico Tronci, editors, *Correct Hardware Design and Verification Methods*, pages 111–125, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[24] Hana Chockler, Orna Kupferman, and Moshe Y. Vardi. Coverage metrics for temporal logic model checking*. *Formal Methods in System Design*, 28:189–212, 2006.

[25] Søren Christensen and Kjeld H. Mortensen. Design/cpn ask-ctl manual. Technical report, PwC, 1996.

[26] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Introduction*, pages 1–11. MIT Press, Cambridge, MA, USA, 2000.

[27] Edmund M. Clarke, Thomas A. Henzinger, and Helmut Veith. *Introduction to Model Checking*, pages 1–26. Springer International Publishing, Cham, 2018.

[28] FIWARE Community. Fiware. Available: `https://www.fiware.org/`. Accessed: 2019/02/01.

[29] FIWARE Community. Fiware-orion. Available: `https://fiware-orion.readthedocs.io/en/master/`. Accessed: 2019/02/01.

[30] FIWARE Community. Fiware-pep-proxy. Available: `http://fiware-pep-proxy.readthedocs.io/en/latest/user_guide/`. Accessed: 2019/02/01.

[31] FIWARE Community. Keyrock. Available: `http://fiware-idm.readthedocs.io/en/latest/introduction.html`. Accessed: 2019/02/01.

[32] FIWARE Community. The fiware catalogue. Available: `https://catalogue.fiware.org/`. Accessed: 2019/02/01.

[33] H. Dai and K. Chen. Optz: a hardware isolation architecture of multi-tasks based on trustzone support. In *IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, pages 391–395, 2017.

[34] S. Demesie Yalew, P. Mendonca, G. Q. Maguire, S. Haridi, and M. Correia. Truapp: A trustzone-based authenticity detection service for mobile apps. In *IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications*, Oct 2017.

[35] Günther Eibl and Dominik Engel. Influence of Data Granularity on Nonintrusive Appliance Load Monitoring. In *Proc. of the 2nd ACM Workshop on Information Hiding and Multimedia Security*, IHMMSec '14. ACM, 2014.

[36] Wei Feng, Dengguo Feng, Ge Wei, Yu Qin, Qianying Zhang, and Dexian Chang. Teem: A user-oriented trusted mobile device for multi-platform security applications.

In Michael Huth, N. Asokan, Srdjan Čapkun, Ivan Flechais, and Lizzie Coles-Kemp, editors, *Trust and Trustworthy Computing*, pages 133–141, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[37] Xianglong Feng, Mengmei Ye, Viswanathan Swaminathan, and Sheng Wei. Towards the security of motion detection-based video surveillance on iot devices. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, page 228–235, Mountain View, CA, USA, 2017. ACM.

[38] Paul Fremantle and Benjamin Aziz. Deriving event data sharing in iot systems using formal modelling and analysis. *Internet of Things*, 8:100092, 2019.

[39] Josh Fruhlinger. The Mirai botnet explained: How teen scammers and CCTV cameras almost brought down the internet. Available: https://www.csoonline.com/article/3258748/security/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html. Accessed: 2019-01-13.

[40] Arik Gabbai. Kevin Ashton Describes "the Internet of Things". Available: https://www.smithsonianmag.com/innovation/kevin-ashton-describes-the-internet-of-things-180953749/, 2015. Accessed: 2019-01-07.

[41] L. Guan, C. Cao, P. Liu, X. Xing, X. Ge, S. Zhang, M. Yu, and T. Jaeger. Building a trustworthy execution environment to defeat exploits from both cyber space and physical space for arm. *IEEE Transactions on Dependable and Secure Computing*, 16(3), May 2019.

[42] Le Guan, Peng Liu, Xinyu Xing, Xinyang Ge, Shengzhi Zhang, Meng Yu, and Trent Jaeger. Trustshadow: Secure execution of unmodified applications with arm trustzone. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, page 488–501, Niagara Falls, NY, USA, 2017. ACM.

[43] Le Guan, Peng Liu, Xinyu Xing, Xinyang Ge, Shengzhi Zhang, Meng Yu, and Trent Jaeger. TrustShadow: Secure Execution of Unmodified Applications with ARM Trust-

Zone. In *Proc. of the 15th Annual International Conf. on Mobile Systems, Applications, and Services*, MobiSys '17. ACM, 2017.

[44] Barbara Guttman and Edward A. Roback. Sp 800-12. an introduction to computer security: The nist handbook. Technical report, Gaithersburg, MD, USA, 1995.

[45] Martin Henze, Lars Hermerschmidt, Daniel Kerpen, Roger Häußling, Bernhard Rumpe, and Klaus Wehrle. A Comprehensive Approach to Privacy in the Cloud-based Internet of Things. *Future Gener. Comput. Syst.*, 56(C):701–718, March 2016.

[46] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on Fog Computing. *J. Netw. Comput. Appl.*, 98(C):27–42, November 2017.

[47] Intel. Intel® Software Guard Extensions (Intel® SGX). Available: `https://software.intel.com/en-us/sgx`. Accessed: 2019-01-13.

[48] Intel. Intel® Software Guard Extensions (Intel® SGX). Available: `https://software.intel.com/sgx/details`. Accessed: 2019-01-13.

[49] S. Jaidka, S. Reeves, and J. Bowen. A coloured petri net approach to model and analyze safety-critical interactive systems. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 347–354, Dec 2019.

[50] Kurt Jensen. *Formal Definition of Coloured Petri Nets*, pages 65–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[51] Kurt Jensen and Lars M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[52] Kurt Jensen and Lars Michael Kristensen. Colored petri nets: a graphical language for formal modeling and validation of concurrent systems. *Commun. ACM*, 58(6):61–70, 2015.

[53] Hang Jiang, Rui Chang, Lu Ren, Weiyu Dong, Liehui Jiang, and Shuiqiao Yang. *An Effective Authentication for Client Application Using ARM TrustZone*, pages 802–813. 12 2017.

[54] S. W. Kim, C. Lee, M. Jeon, H. Y. Kwon, H. W. Lee, and C. Yoo. Secure device access for automotive software. In *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 177–181, 2013.

[55] Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, and Daniel Fitton. Smart Objects as Building Blocks for the Internet of Things. *IEEE Internet Computing*, 14(1):44–51, January 2010.

[56] Vaibhav Kulkarni, Bertil Chapuis, and Benoît Garbinato. Privacy-preserving location-based services by using intel sgx. In *Proceedings of the First International Workshop on Human-Centered Sensing, Networking, and Systems*, page 13–18, Delft, Netherlands, 2017. ACM.

[57] Ashalatha Kunnappilly, Raluca Marinescu, and Cristina Seceleanu. A model-checking-based framework for analyzing ambient assisted living solutions. *Sensors*, 19(22):5057, Nov 2019.

[58] Markku Kylanpaa and Aarne Rantala. Remote attestation for embedded systems. volume 9588, pages 79–92, 06 2016.

[59] Daniel Le Métayer. Privacy by design: A formal framework for the analysis of architectural choices. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy*, CODASPY '13, pages 95–104, New York, NY, USA, 2013. ACM.

[60] Kristen Lee. Despite risks, healthcare IT professionals stick with mobile. Available: https://searchhealthit.techtarget.com/feature/Despite-risks-healthcare-IT-professionals-stick-with-mobile. Accessed: 2018-12-03.

[61] Sungbum Lee and Jong-Hyouk Lee. Tee based session key establishment protocol for secure infotainment systems. *Design Automation for Embedded Systems*, 22:215–224, 2018.

[62] Yun-kyung Lee, Jeongnyeo Kim, Kyung-Soo Lim, and Hyunsoo Yoon. Secure mobile device structure for trust iot. *The Journal of Supercomputing*, 74, 10 2017.

[63] Vincent Leest, Roel Maes, Geert-Jan Schrijen, and Pim Tuyls. *Hardware Intrinsic Security to Protect Value in the Mobile Market*, pages 188–198. 01 2014.

[64] C. Lesjak, D. Hein, and J. Winter. Hardware-security technologies for industrial iot: Trustzone and security controller. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pages 002589–002595, Nov 2015.

[65] Fagen Li, Zhaohui Zheng, and Chunhua Jin. Secure and Efficient Data Transmission in the Internet of Things. *Telecommunication Systems*, 62(1):111–122, May 2016.

[66] Xueping Liang, Sachin Shetty, Juan Zhao, Daniel Bowden, Danyi Li, and Jihong Liu. Towards decentralized accountability and self-sovereignty in healthcare systems. In *Proceedings of the International Conference on Information and Communications Security*, 12 2017.

[67] Xueping Liang, Sachin Shetty, Juan Zhao, Daniel Bowden, Danyi Li, and Jihong Liu. Towards decentralized accountability and self-sovereignty in healthcare systems. In Sihan Qing, Chris Mitchell, Liqun Chen, and Dongmei Liu, editors, *Information and Communications Security*, pages 387–398, Cham, 2018. Springer International Publishing.

[68] Z. Liu and J. Liu. Formal verification of blockchain smart contract based on colored petri net models. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 555–560, Jul 2019.

[69] Konstantinos Markantonakis, Raja Naeem Akram, and Mehari G. Msgna. Secure and trusted application execution on embedded devices. In Ion Bica, David Naccache, and Emil Simion, editors, *International Conference for Information Technology and Communications*, pages 3–24, Bucharest, Romania, 2015. Springer.

[70] Stephen McLaughlin, Patrick McDaniel, and William Aiello. Protecting Consumer Privacy from Electric Load Monitoring. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 87–98, New York, NY, USA, 2011. ACM.

[71] Mitar Milutinovic, Warren He, Howard Wu, and Maxinder Kanwal. Proof of luck: An efficient blockchain consensus protocol. In *Proceedings of the 1st Workshop on System Software for Trusted Execution*, SysTEX '16, New York, NY, USA, 2016. Association for Computing Machinery.

[72] Saeed Mirzamohammadi, Justin A. Chen, Ardalan Amiri Sani, Sharad Mehrotra, and Gene Tsudik. Ditio: Trustworthy auditing of sensor activities in mobile & iot devices. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, Delft, Netherlands, 2017. ACM.

[73] Susan Moore and Emma Keen. Gartner Forecasts Worldwide Information Security Spending to Exceed $124 Billion in 2019. Available: https://www.gartner.com/en/newsroom/press-releases/2018-08-15-gartner-forecasts-worldwide-information-security-spending-to-exceed-124-billion-in-2019. Accessed: 2019-01-13.

[74] H. Nguyen, R. Ivanov, L. T. X. Phan, O. Sokolsky, J. Weimer, and I. Lee. Logsafe: Secure and scalable data logger for iot devices. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 141–152, April 2018.

[75] Oracle. XACML Policy Enforcement Point. Available: `https://docs.oracle.com/cd/E27515_01/common/tutorials/authz_xacml_pep.html`. Accessed: 2019/02/01.

[76] André Osterhues, Ahmad-Reza Sadeghi, Marko Wolf, Christian Stüble, and N. Asokan. Securing peer-to-peer distributions for mobile devices. In Liqun Chen, Yi Mu, and Willy Susilo, editors, *International Conference on Information Security Practice and Experience*, pages 161–175, Sydney, NSW, Australia, 2008. Springer.

[77] J. Park and Kwangjo Kim. Tm-coin: Trustworthy management of tcb measurements in iot. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 654–659, March 2017.

[78] Andrew J. Paverd and Andrew P. Martin. Hardware security for device authentication in the smart grid. In Jorge Cuellar, editor, *Smart Grid Securit. SmartGridSec 2012.y*, pages 72–84, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[79] Travis Peters, Reshma Lal, Srikanth Varadarajan, Pradeep Pappachan, and David Kotz. BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX. In *Proc. of the Intl Workshop on Hardware and Architectural Support for Security and Privacy*, New York, NY, USA, 2018. ACM.

[80] Robert Pettersen., Håvard D. Johansen., and Dag Johansen. Secure edge computing with arm trustzone. In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security*, pages 102–109, Porto, Portugal, 2017. INSTICC, SciTePress.

[81] S. Pinto, T. Gomes, J. Pereira, J. Cabral, and A. Tavares. Iioteed: An enhanced, trusted execution environment for industrial iot edge devices. *IEEE Internet Computing*, 21(1):40–47, 2017.

[82] Sandro Pinto, Jorge Pereira, Tiago Gomes, Adriano Tavares, and Jorge Cabral. Ltzvisor: Trustzone is the key. In *Proceedings of the 29th Euromicro Conference on Real-Time Systems (ECRTS)*, 06 2017.

[83] Rafael Pires, Marcelo Pasin, Pascal Felber, and Christof Fetzer. Secure content-based routing using intel software guard extensions. In *Proc. of the 17th International Middleware Conf.*, Middleware '16. ACM, 2016.

[84] Yu Qin, Yingjun Zhang, and Wei Feng. Tics: Trusted industry control system based on hardware security module. pages 485–493, 10 2017.

[85] Vincent Raes, Jan Vossaert, and Vincent Naessens. *Development of an Embedded Platform for Secure CPS Services*, pages 19–34. Springer, Cham, 01 2018.

[86] Alison DeNisco Rayome. DDoS attacks increased 91% in 2017 thanks to IoT. Available: https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/. Accessed: 2019-01-13.

[87] Alejandro Rodríguez, Lars Michael Kristensen, and Adrian Rutle. Formal modelling and incremental verification of the MQTT iot protocol. *Trans. Petri Nets Other Model. Concurr.*, 14:126–145, 2019.

[88] Alejandro Rodríguez, Lars Michael Kristensen, and Adrian Rutle. On CTL model checking of the MQTT iot protocol using the sweep-line method. In *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE 2019), co-located with the 40th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2019 and the 19th International Conference on Application of Concurrency to System Design ACSD 2019 and the 1st IEEE International Conference on Process Mining Process Mining 2019, Aachen, Germany, June 23-28, 2019*, pages 57–72, 2019.

[89] Margaret Rouse and Shaun Sutner. PHI breach (protected health information breach). Available: https://searchhealthit.techtarget.com/definition/PHI-breach-protected-health-information-breach. Accessed: 2018-12-03.

[90] M. Sabt, M. Achemlal, and A. Bouabdallah. Trusted execution environment: What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64, Aug 2015.

[91] Lilia Sampaio, Fábio Silva, Amanda Souza, Andrey Brito, and Pascal Felber. Secure and privacy-aware data dissemination for cloud-based applications. In *Proceedings of the 10th International Conference on Utility and Cloud Computing*, page 47–56, New York, NY, USA, 2017. ACM.

[92] Steffen Schulz, André Schaller, Florian Kohnhäuser, and Stefan Katzenbeisser. Boot attestation: Secure remote reporting with off-the-shelf iot sensors. pages 437–455, 08 2017.

[93] Younes Seifi, Suriadi Suriadi, Ernest Foo, and Colin Boyd. Analysis of object-specific authorization protocol (osap) using coloured petri nets. In *Proceedings of the Tenth Australasian Information Security Conference - Volume 125*, AISC '12, page 47–58, AUS, 2012. Australian Computer Society, Inc.

[94] A. Shaghaghi, M. A. Kaafar, S. Scott-Hayward, S. S. Kanhere, and S. Jha. Towards Policy Enforcement Point as a Service (PEPS). In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 50–55, Nov 2016.

[95] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. Emlog: Tamper-resistant system logging for constrained devices with tees. 12 2017.

[96] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. Establishing mutually trusted channels for remote sensing devices with trusted execution environments. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, Reggio Calabria, Italy, 2017. ACM.

[97] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3, 2016.

[98] SSL Shopper. What is SSL? SSL Certificate Basics. Available: https://www.sslshopper.com/what-is-ssl.html. Accessed: 2019-01-03.

[99] F. Siddiqui, M. Hagan, and S. Sezer. Embedded policing and policy enforcement approach for future secure iot technologies. In *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, pages 1–10, London, UK, 2018. IEEE.

[100] Leandro Silva, Pedro Barbosa, Rodolfo Silva, and Andrey Brito. Security and privacy aware data aggregation on cloud computing. *Journal of Internet Services and Applications*, 9, 04 2018.

[101] Leandro Ventura Silva, Rodolfo Marinho, Jose Luis Vivas, and Andrey Brito. Security and privacy preserving data aggregation in cloud computing. In *Proc. of the Symposium on Applied Computing*, SAC '17. ACM, 2017.

[102] He Sun, Kun Sun, Yuewu Wang, Jiwu Jing, and Sushil Jajodia. Trustdump: Reliable memory acquisition on smartphones. In Mirosław Kutyłowski and Jaideep Vaidya, editors, *Computer Security, ESORICS 2014. Lecture Notes in Computer Science*, volume 8712, pages 202–218. Springer International Publishing, 09 2014.

[103] Suriadi Suriadi, Chun Ouyang, and Ernest Foo. Privacy compliance verification in cryptographic protocols. In Kurt Jensen, Wil M. van der Aalst, Marco Ajmone Marsan, Giuliana Franceschinis, Jetty Kleijn, and Lars Michael Kristensen, editors, *Transactions on Petri Nets and Other Models of Concurrency VI*, pages 251–276. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[104] Azeria team. Trustonic's Kinibi TEE Implementation. Available: `https://azeria-labs.com/trustonics-kinibi-tee-implementation/`. Accessed: 2020-01-13.

[105] Mesalock team. Rust OP-TEE TrustZone SDK. Available: `https://github.com/mesalock-linux/rust-optee-trustzone-sdk`. Accessed: 2020-01-13.

[106] OP-TEE team. OP-TEE Trusted OS. Available: `https://github.com/OP-TEE/optee_os`. Accessed: 2020-01-13.

[107] Scone team. Scone. Available: `https://scontain.com/index.html?lang=en`. Accessed: 2020-01-13.

[108] H. Tsunoda and G. M. Keeni. Feasibility of societal model for securing internet of things. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 541–546, Valencia, Spain, 2017. IEEE.

[109] D. C. G. Valadares, M. S. L. da Silva, A. M. E. Brito, and E. M. Salvador. Achieving Data Dissemination with Security using FIWARE and Intel Software Guard Extensions (SGX). In *IEEE Symposium on Computers and Communications*, 2018.

[110] Rob van der Meulen. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. Available: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016. Accessed: 2019-01-13.

[111] Rini van Solingen (Revision), Vic Basili (Original article, 1994 ed.), Gianluigi

Caldiera (Original article, 1994 ed.), and H. Dieter Rombach (Original article, 1994 ed.). *Goal Question Metric (GQM) Approach*. American Cancer Society, 2002.

[112] Emília Villani, Rodrigo Pastl Pontes, Guilherme Kisselofl Coracini, and Ana Maria Ambrósio. Integrating model checking and model based testing for industrial software development. *Computers in Industry*, 104:88 – 102, 2019.

[113] Christian Wachsmann, Liqun Chen, Kurt Dietrich, Hans Löhr, Ahmad-Reza Sadeghi, and Johannes Winter. Lightweight anonymous authentication with tls and daa for embedded mobile devices. In Mike Burmester, Gene Tsudik, Spyros Magliveras, and Ivana Ilić, editors, *Information Security. AINA 2019*, pages 84–98, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[114] Muhammad Wahab, Pascal Cotret, Mounir Nasr Allah, Guillaume Hiet, Vianney Lapotre, and Guy Gogniat. Armhex: A hardware extension for dift on arm-based socs. pages 1–7, 09 2017.

[115] J. Wang, Z. Hong, Y. Zhang, and Y. Jin. Enabling security-enhanced attestation with intel sgx for remote terminal and iot. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):88–96, Jan 2018.

[116] Rui Wang, Lars Michael Kristensen, Hein Meling, and Volker Stolz. Automated test case generation for the paxos single-decree protocol using a coloured petri net model. *Journal of Logical and Algebraic Methods in Programming*, 104:254 – 273, 2019.

[117] Stephanie Weagle. The Rise of IoT Botnet Threats and DDoS attacks. Available: https://www.corero.com/blog/870-the-rise-of-iot-botnet-threats-and-ddos-attacks.html. Accessed: 2019-01-13.

[118] Rolf Weber. Internet of Things: Privacy Issues Revisited. *Computer Law and Security Review*, 31, 2015.

[119] Jorge Werner, Carla Merkle Westphall, and Carlos Westphall. Cloud Identity Management: A Survey on Privacy Strategies. *Computer Networks*, 122, 2017.

[120] Michael Westergaard, Sami Evangelista, and Lars Michael Kristensen. Asap: An extensible platform for state space analysis. In Giuliana Franceschinis and Karsten Wolf, editors, *Applications and Theory of Petri Nets*, pages 303–312, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[121] Judson Wilson, Riad S. Wahby, Henry Corrigan-Gibbs, Dan Boneh, Philip Levis, and Keith Winstein. Trust but verify: Auditing the secure internet of things. In *Proc. of the 15th Annual International Conf. on Mobile Systems, Applications, and Services*, MobiSys '17. ACM, 2017.

[122] H. Wirtz, T. Zimmermann, M. Ceriotti, and K. Wehrle. Encrypting data to pervasive contexts. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 309–315, 2017.

[123] T. Xu, D. Zhu, and S. Yao. Model checking for rare-event in control logical petri nets based on importance sampling. *IEEE Access*, pages 1–1, 2020.

[124] Anil Yadav, Nitin Rakesh, Sujata Pandey, and Rajat Singh. Iotee-an integrated framework for rapid trusted iot application development. pages 1829–1834, 05 2016.

[125] S. D. Yalew, G. Q. Maguire, S. Haridi, and M. Correia. T2droid: A trustzone-based dynamic analyser for android applications. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 240–247, Sydney, NSW, Australia, 2017. IEEE.

[126] Hongyang Yan, Xuan Li, Yu Wang, and Chunfu Jia. Centralized duplicate removal video storage system with privacy preservation in iot. *Sensors*, 18(6):1814, Jun 2018.

[127] X. Yan-ling, P. Wei, and Z. Xin-guo. Design and implementation of secure embedded systems based on trustzone. In *2008 International Conference on Embedded Software and Systems*, pages 136–141, 2008.

[128] Xia Yang, Peng Shi, Bo Tian, Bing Zeng, and Wei Xiao. Trust-e: A trusted embedded operating system based on the arm trustzone. In *IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing, and 11th Intl Conf on Autonomic and Trusted Computing, and 14th Intl Conf on Scalable Computing and Communications*, pages 495–501, 12 2014.

[129] Benjamin Yankson, Farkhund Iqbal, Zhihui Lu, Xiaoling Wang, and Patrick C. K. Hung. Modeling privacy preservation in smart connected toys by petri-nets. In *52nd Hawaii International Conference on System Sciences*, 2019.

[130] N. Zhang, H. Sun, K. Sun, W. Lou, and Y. T. Hou. Cachekit: Evading memory introspection using cache incoherence. In *IEEE European Symposium on Security and Privacy (EuroS P)*, March 2016.

[131] N. Zhang, K. Sun, W. Lou, and Y. T. Hou. Case: Cache-assisted secure execution on arm processors. In *IEEE Symposium on Security and Privacy (SP)*, pages 72–90, 2016.

[132] Ning Zhang, Jin Li, Wenjing Lou, and Y. Hou. *PrivacyGuard: Enforcing Private Data Usage with Blockchain and Attested Execution: ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings*, pages 345–353. 09 2018.

[133] Xiaosong Zhang, Tan yu an, Yuan Xue, Quanxin Zhang, Yuanzhang Li, Can Zhang, and Jun Zheng. Cryptographic key protection against frost for mobile devices. *Cluster Computing*, 20, 09 2017.

[134] B. Zhao, Y. Xiao, Y. Huang, and X. Cui. A private user data protection mechanism in trustzone architecture based on identity authentication. *Tsinghua Science and Technology*, 22(2):218–225, 2017.

[135] X. Zheng, Y. He, J. Ma, G. Shi, and D. Meng. Tz-kpm:kernel protection mechanism on embedded devices on hardware-assisted isolated environment. In *IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 663–670, 2016.

Appendix A

Aditional Information regarding the Systematic Literature Review

A.1 Selection Phase information

Next, the search string for each scientific repository as well as the number of documents resulted are shown.

EI Compendex: ((("Internet of Things" OR "Edge Computing" OR "Edge of Things" OR "Embedded Systems" OR "EoT" OR "Internet of Everything" OR "IoE" OR "IoT" OR "Web of Things" OR "WoT") AND ("Trusted Execution Environment" OR "Intel SGX" OR "Keystone Enclave" OR "Sanctum" OR "SGX" OR "Software Guard Extensions" OR "TEE" OR "Trusted Execution Technology" OR "Trustzone") AND ("Advantage" OR "Benefit" OR " Positive Point " OR "Disadvantage" OR "Drawback" OR " Negative Point " OR "Security" OR "Confidentiality" OR "Integrity" OR "Privacy" OR "Protection" OR "Trustworthiness" OR "Solution" OR "Approach" OR "Framework" OR "Mechanism" OR "Method" OR "Technique" OR "Tool")) WN ALL) + ((ca OR ja OR cp OR ip) WN DT) AND (english WN LA)

Link (92 results) `https://www.engineeringvillage.com/search/expert.url?SEARCHID=0eb33193M9322M4898M98ffM36770aa68c5c&COUNT=1&usageOrigin=&usageZone=`

Scopus: (TITLE-ABS-KEY (("Internet of Things" OR "Edge Computing" OR "Edge of Things" OR "Embedded Systems" OR "EoT" OR "Internet of Everything" OR "IoE" OR

"IoT" OR "Web of Things" OR "WoT")) AND TITLE-ABS-KEY (("Trusted Execution Environment" OR "Intel SGX" OR "Keystone Enclave" OR "Sanctum" OR "SGX" OR "Software Guard Extensions" OR "TEE" OR "Trusted Execution Technology" OR "Trustzone")) AND TITLE-ABS-KEY (("Advantage" OR "Benefit" OR " Positive Point " OR "Disadvantage" OR "Drawback" OR " Negative Point " OR "Security" OR "Confidentiality" OR "Integrity" OR "Privacy" OR "Protection" OR "Trustworthiness" OR "Solution" OR "Approach" OR "Framework"))) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (SRCTYPE , "p") OR LIMIT-TO (SRCTYPE , "j"))

Link (73 results): https://www.scopus.com/results/results.uri?sort=plf-f&src=s&st1=%28%22Internet+of+Things%22+OR+%22Edge+Computing%22+OR+%22Edge+of+Things%22+OR+%22Embedded+Systems%22+OR+%22EoT%22+OR+%22Internet+of+Everything%22+OR+%22IoE%22+OR+%22IoT%22+OR+%22Web+of+Things%22+OR+%22WoT%22%29&st2=%28%22Trusted+Execution+Environment%22+OR+%22Intel+SGX%22+OR+%22Keystone+Enclave%22+OR+%22Sanctum%22+OR+%22SGX%22+OR+%22Software+Guard+Extensions%22+OR+%22TEE%22+OR+%22Trusted+Execution+Technology%22+OR+%22Trustzone%22%29&nlo=&nlr=&nls=&sid=d25bb4ccf398b725600ff360a9f90cf0&sot=b&sdt=cl&cluster=scosubtype%2c%22cp%22%2ct%2c%22ar%22%2ct%2bscolang%2c%22English%22%2ct%2bscosrctype%2c%22p%22%2ct%2c%22j%22%2ct&sl=649&s=%28TITLE-ABS-KEY%28%28%22Internet+of+Things%22+OR+%22Edge+Computing%22+OR+%22Edge+of+Things%22+OR+%22Embedded+Systems%22+OR+%22EoT%22+OR+%22Internet+of+Everything%22+OR+%22IoE%22+OR+%22IoT%22+OR+%22Web+of+Things%22+OR+%22WoT%22%29%29+AND+TITLE-ABS-KEY%28%28%22Trusted+Execution+Environment%22+OR+%22Intel+SGX%22+OR+%22Keystone+Enclave%22+OR+%22Sanctum%22+OR+%22SGX%22+OR+%22Software+Guard+Extensions%22+OR+%22TEE%22+OR+%22Trusted+Execution+Technology%22+OR+%22Trustzone%22%29%29AND+TITLE-ABS-KEY%28%28%22Advantage%22+OR+%22Benefit%22+OR+%22+Positive+Point+

%22+OR+%22Disadvantage%22+OR+%22Drawback%22+OR+%22+Negative+
Point+%22+OR+%22Security%22+OR+%22Confidentiality%22+OR+
%22Integrity%22+OR+%22Privacy%22+OR+%22Protection%22+OR+
%22Trustworthiness%22+OR+%22Solution%22+OR+%22Approach%22+OR+
%22Framework%22%29%29%29&origin=resultslist&zone=leftSideBar&
editSaveSearch=&txGid=3166f0cb4bbd284ed09f14783dc62b33

(TITLE-ABS-KEY (("Internet of Things" OR "Edge Computing" OR "Edge of Things" OR "Embedded Systems" OR "EoT" OR "Internet of Everything" OR "IoE" OR "IoT" OR "Web of Things" OR "WoT")) AND TITLE-ABS-KEY (("Trusted Execution Environment" OR "Intel SGX" OR "Keystone Enclave" OR "Sanctum" OR "SGX" OR "Software Guard Extensions" OR "TEE" OR "Trusted Execution Technology" OR "Trustzone")) AND TITLE-ABS-KEY (("Mechanism" OR "Method" OR "Technique" OR "Tool"))) AND (LIMIT-TO (SRCTYPE , "p") OR LIMIT-TO (SRCTYPE , "j")) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar")) AND (LIMIT-TO (LANGUAGE , "English"))

Link (32 results): https://www.scopus.com/results/results.uri?
sort=plf-f&src=s&sid=3edf63535b624994f1eef45446cb9cd8&sot=
a&sdt=a&cluster=scosrctype%2c%22p%22%2ct%2c%22j%22%2ct%
2bscosubtype%2c%22cp%22%2ct%2c%22ar%22%2ct%2bscolang%2c%
22English%22%2ct&sl=454&s=%28TITLE-ABS-KEY%28%28%22Internet+
of+Things%22+OR+%22Edge+Computing%22+OR+%22Edge+of+Things%
22+OR+%22Embedded+Systems%22+OR+%22EoT%22+OR+%22Internet+of+
Everything%22+OR+%22IoE%22+OR+%22IoT%22+OR+%22Web+of+Things%
22+OR+%22WoT%22%29%29+AND+TITLE-ABS-KEY%28%28%22Trusted+
Execution+Environment%22+OR+%22Intel+SGX%22+OR+%22Keystone+
Enclave%22+OR+%22Sanctum%22+OR+%22SGX%22+OR+%22Software+
Guard+Extensions%22+OR+%22TEE%22+OR+%22Trusted+Execution+
Technology%22+OR+%22Trustzone%22%29%29AND+TITLE-ABS-KEY%
28%28%22Mechanism%22+OR+%22Method%22+OR+%22Technique%22+OR+
%22Tool%22%29%29%29&origin=searchadvanced&editSaveSearch=
&txGid=f77cf42f19c4d5b5e1e4fccbd79a953c

Wiley Library: ("Internet of Things" OR "Edge Computing" OR "Edge of Things" OR "Embedded Systems" OR "EoT" OR "Internet of Everything" OR "IoE" OR "IoT" OR "Web of Things" OR "WoT") AND ("Trusted Execution Environment" OR "Intel SGX" OR "Keystone Enclave" OR "Sanctum" OR "SGX" OR "Software Guard Extensions" OR "TEE" OR "Trusted Execution Technology" OR "Trustzone") AND ("Advantage" OR "Benefit" OR " Positive Point " OR "Disadvantage" OR "Drawback" OR " Negative Point " OR "Security" OR "Confidentiality" OR "Integrity" OR "Privacy" OR "Protection" OR "Trustworthiness" OR "Solution" OR "Approach" OR "Framework" OR "Mechanism" OR "Method" OR "Technique" OR "Tool")

Link (7 results): `https://onlinelibrary.wiley.com/action/doSearch?AllField=%28%22Internet+of+Things%22+OR+%22Edge+Computing%22+OR+%22Edge+of+Things%22+OR+%22Embedded+Systems%22+OR+%22EoT%22+OR+%22Internet+of+Everything%22+OR+%22IoE%22+OR+%22IoT%22+OR+%22Web+of+Things%22+OR+%22WoT%22%29+AND+%28%22Trusted+Execution+Environment%22+OR+%22Intel+SGX%22+OR+%22Keystone+Enclave%22+OR+%22Sanctum%22+OR+%22SGX%22+OR+%22Software+Guard+Extensions%22+OR+%22TEE%22+OR+%22Trusted+Execution+Technology%22+OR+%22Trustzone%22%29+AND+%28%22Advantage%22+OR+%22Benefit%22+OR+%22+Positive+Point+%22+OR+%22Disadvantage%22+OR+%22Drawback%22+OR+%22+Negative+Point+%22+OR+%22Security%22+OR+%22Confidentiality%22+OR+%22Integrity%22+OR+%22Privacy%22+OR+%22Protection%22+OR+%22Trustworthiness%22+OR+%22Solution%22+OR+%22Approach%22+OR+%22Framework%22+OR+%22Mechanism%22+OR+%22Method%22+OR+%22Technique%22+OR+%22Tool%22%29&ConceptID=68&PubType=journal&content=articlesChapters&countTerms=true&target=default&AfterYear=2000&BeforeYear=2018`

IEEExplore: ("Internet of Things" OR "Edge Computing" OR "Edge of Things" OR "Embedded Systems" OR "EoT" OR "Internet of Everything" OR "IoE" OR "IoT" OR "Web of Things" OR "WoT") AND ("Trusted Execution Environment" OR "Intel SGX" OR "Keystone Enclave" OR "Sanctum" OR "SGX" OR "Software Guard Extensions" OR "TEE" OR

"Trusted Execution Technology" OR "Trustzone") AND ("Advantage" OR "Benefit" OR " Positive Point " OR "Disadvantage" OR "Drawback" OR " Negative Point " OR "Security" OR "Confidentiality" OR "Integrity" OR "Privacy" OR "Protection" OR "Trustworthiness" OR "Solution" OR "Approach" OR "Framework" OR "Mechanism" OR "Method" OR "Technique" OR "Tool")

Link (56 results): `https://ieeexplore.ieee.org/search/searchresult.jsp?queryText=(%22Internet%20of%20Things%22%20OR%20%22Edge%20Computing%22%20OR%20%22Edge%20of%20Things%22%20OR%20%22Embedded%20Systems%22%20OR%20%22EoT%22%20OR%20%22Internet%20of%20Everything%22%20OR%20%22IoE%22%20OR%20%22IoT%22%20OR%20%22Web%20of%20Things%22%20OR%20%22WoT%22)%20AND%20(%22Trusted%20Execution%20Environment%22%20OR%20%22Intel%20SGX%22%20OR%20%22Keystone%20Enclave%22%20OR%20%22Sanctum%22%20OR%20%22SGX%22%20OR%20%22Software%20Guard%20Extensions%22%20OR%20%22TEE%22%20OR%20%22Trusted%20Execution%20Technology%22%20OR%20%22Trustzone%22)%20AND%20(%22Advantage%22%20OR%20%22Benefit%22%20OR%20%22%20Positive%20Point%20%22%20OR%20%22Disadvantage%22%20OR%20%22Drawback%22%20OR%20%22%20Negative%20Point%20%22%20OR%20%22Security%22%20OR%20%22Confidentiality%22%20OR%20%22Integrity%22%20OR%20%22Privacy%22%20OR%20%22Protection%22%20OR%20%22Trustworthiness%22%20OR%20%22Solution%22%20OR%20%22Approach%22%20OR%20%22Framework%22%20OR%20%22Mechanism%22%20OR%20%22Method%22%20OR%20%22Technique%22%20OR%20%22Tool%22)&highlight=true&returnFacets=ALL&returnType=SEARCH&refinements=ContentType:Conferences&refinements=ContentType:Journals%20.AND.%20Magazines`

ACM: ("Internet of Things" OR "Edge Computing" OR "Edge of Things" OR "Embedded Systems" OR "EoT" OR "Internet of Everything" OR "IoE" OR "IoT" OR "Web of Things" OR "WoT") AND ("Trusted Execution Environment" OR "Intel SGX" OR "Keystone Enclave" OR "Sanctum" OR "SGX" OR "Software Guard Extensions" OR "TEE" OR

"Trusted Execution Technology" OR "Trustzone") AND ("Advantage" OR "Benefit" OR " Positive Point " OR "Disadvantage" OR "Drawback" OR " Negative Point " OR "Security" OR "Confidentiality" OR "Integrity" OR "Privacy" OR "Protection" OR "Trustworthiness" OR "Solution" OR "Approach" OR "Framework" OR "Mechanism" OR "Method" OR "Technique" OR "Tool")

Link (43 results): `https://dl.acm.org/results.cfm?query=%28%22Internet%20of%20Things%22%20OR%20%22Edge%20Computing%22%20OR%20%22Edge%20of%20Things%22%20OR%20%22Embedded%20Systems%22%20OR%20%22EoT%22%20OR%20%22Internet%20of%20Everything%22%20OR%20%22IoE%22%20OR%20%22IoT%22%20OR%20%22Web%20of%20Things%22%20OR%20%22WoT%22%29%20AND%20%28%22Trusted%20Execution%20Environment%22%20OR%20%22Intel%20SGX%22%20OR%20%22Keystone%20Enclave%22%20OR%20%22Sanctum%22%20OR%20%22SGX%22%20OR%20%22Software%20Guard%20Extensions%22%20OR%20%22TEE%22%20OR%20%22Trusted%20Execution%20Technology%22%20OR%20%22Trustzone%22%29%20AND%20%28%22Advantage%22%20OR%20%22Benefit%22%20OR%20%22%20Positive%20Point%20%22%20OR%20%22Disadvantage%22%20OR%20%22Drawback%22%20OR%20%22%20Negative%20Point%20%22%20OR%20%22Security%22%20OR%20%22Confidentiality%22%20OR%20%22Integrity%22%20OR%20%22Privacy%22%20OR%20%22Protection%22%20OR%20%22Trustworthiness%22%20OR%20%22Solution%22%20OR%20%22Approach%22%20OR%20%22Framework%22%20OR%20%22Mechanism%22%20OR%20%22Method%22%20OR%20%22Technique%22%20OR%20%22Tool%22%29&filtered=&within=owners.owner=GUIDE&dte=&bfr=&srt=%5Fscore`

Springer Link: ("Internet of Things" OR "Edge Computing" OR "Edge of Things" OR "Embedded Systems" OR "EoT" OR "Internet of Everything" OR "IoE" OR "IoT" OR "Web of Things" OR "WoT") AND ("Trusted Execution Environment" OR "Intel SGX" OR "Keystone Enclave" OR "Sanctum" OR "SGX" OR "Software Guard Extensions" OR "TEE" OR "Trusted Execution Technology" OR "Trustzone") AND ("Advantage" OR "Benefit" OR " Positive Point " OR "Disadvantage" OR "Drawback" OR " Negative Point " OR "Secu-

rity" OR "Confidentiality" OR "Integrity" OR "Privacy" OR "Protection" OR "Trustworthiness" OR "Solution" OR "Approach" OR "Framework" OR "Mechanism" OR "Method" OR "Technique" OR "Tool")

Links (128 + 289 results): `https://link.springer.com/search?date-facet-mode=between&query=%28%22Internet+of+Things%22+OR+%22Edge+Computing%22+OR+%22Edge+of+Things%22+OR+%22Embedded+Systems%22+OR+%22EoT%22+OR+%22Internet+of+Everything%22+OR+%22IoE%22+OR+%22IoT%22+OR+%22Web+of+Things%22+OR+%22WoT%22%29+AND+%28%22Trusted+Execution+Environment%22+OR+%22Intel+SGX%22+OR+%22Keystone+Enclave%22+OR+%22Sanctum%22+OR+%22SGX%22+OR+%22Software+Guard+Extensions%22+OR+%22TEE%22+OR+%22Trusted+Execution+Technology%22+OR+%22Trustzone%22%29+AND+%28%22Advantage%22+OR+%22Benefit%22+OR+%22+Positive+Point+%22+OR+%22Disadvantage%22+OR+%22Drawback%22+OR+%22+Negative+Point+%22+OR+%22Security%22+OR+%22Confidentiality%22+OR+%22Integrity%22+OR+%22Privacy%22+OR+%22Protection%22+OR+%22Trustworthiness%22+OR+%22Solution%22+OR+%22Approach%22+OR+%22Framework%22+OR+%22Mechanism%22+OR+%22Method%22+OR+%22Technique%22+OR+%22Tool%22%29&facet-start-year=2000&facet-end-year=2018&facet-discipline=%22Computer+Science%22&facet-content-type=%22Article%22`

`https://link.springer.com/search?facet-discipline=%22Computer+Science%22&facet-content-type=%22ConferencePaper%22&query=%28%22Internet+of+Things%22+OR+%22Edge+Computing%22+OR+%22Edge+of+Things%22+OR+%22Embedded+Systems%22+OR+%22EoT%22+OR+%22Internet+of+Everything%22+OR+%22IoE%22+OR+%22IoT%22+OR+%22Web+of+Things%22+OR+%22WoT%22%29+AND+%28%22Trusted+Execution+Environment%22+OR+%22Intel+SGX%22+OR+%22Keystone+Enclave%22+OR+%22Sanctum%22+OR+%22SGX%22+OR+%22Software+Guard+Extensions%22+OR+%22TEE%22+OR+%22Trusted+Execution+Technology%22+OR+%22Trustzone%22%29+AND+%28%`

22Advantage%22+OR+%22Benefit%22+OR+%22+Positive+Point+%22+
OR+%22Disadvantage%22+OR+%22Drawback%22+OR+%22+Negative+
Point+%22+OR+%22Security%22+OR+%22Confidentiality%22+OR+
%22Integrity%22+OR+%22Privacy%22+OR+%22Protection%22+OR+
%22Trustworthiness%22+OR+%22Solution%22+OR+%22Approach%
22+OR+%22Framework%22+OR+%22Mechanism%22+OR+%22Method%
22+OR+%22Technique%22+OR+%22Tool%22%29&date-facet-mode=
between&facet-start-year=2000&previous-start-year=1972&
facet-end-year=2018&previous-end-year=2018

Filters: English, Computers Science, Articles, 2000-2018; English, Computers Science, Conference Paper, 2000-2018.

A.2 General Results

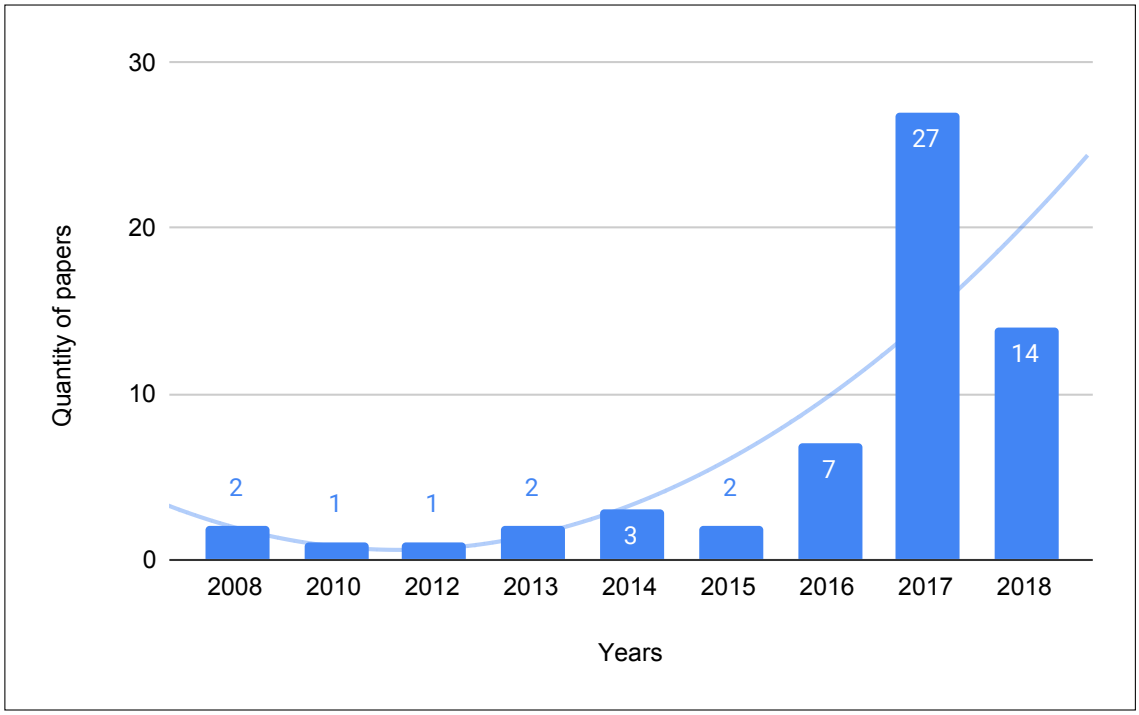


Figure A.1: Quantity of papers by year

As we can notice in Fig. A.1, the number of papers has grown over the years. In 2018 we have just 14 papers because the search was performed at the end of August. We have

to consider also delays at the publishing phase regarding conference and journal papers already accepted during the first semester, but that only becomes available during the second semester. Among these papers, 47 were published in conference, symposium, or workshop proceedings, while 11 were published in journals, as seen in Fig. A.2. The conference names are listed in Table A.2 and journal names are listed in Table A.3. The names with an asterisk * mean that the conference or journal had two papers published.

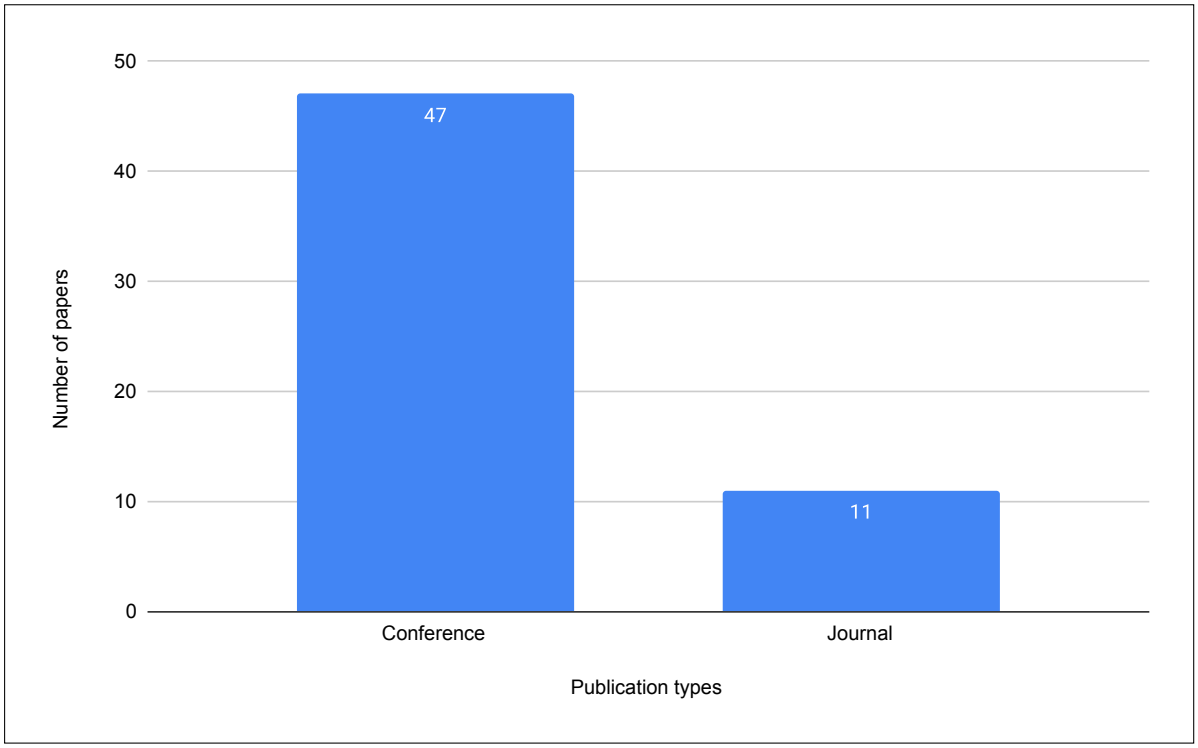


Figure A.2: Number of papers by type

As seen in Fig. A.3, most of the papers are collaboration between 3, 4, 5 or 6 authors. A lower quantity has just 2 or more than 6 authors. With the exception of just 4 works, all the papers present 6 or more pages, as seen in Fig. A.4. The Fig. A.5 presents the number of papers according to the number of citations. As many papers were still recent published when we conducted the search, they have no citations or have just a few (1 to 6). Only a few articles have more than 6 citations.

We present in Fig. A.6 the quality assessment grades and the respective quantity of papers for each grade. According to the quality assessment criteria, we can see that most of the papers got a grade between 5.5 and 8.5. In Fig. A.7, we can see the number of papers regarding the quality classification, which was attributed considering the papers' quality score.

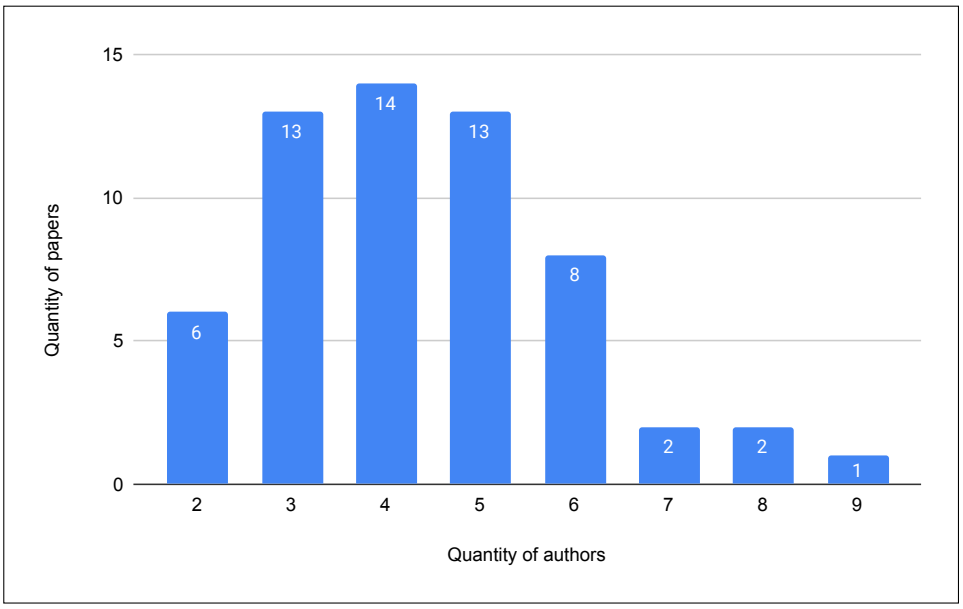


Figure A.3: Quantity of papers by quantity of authors

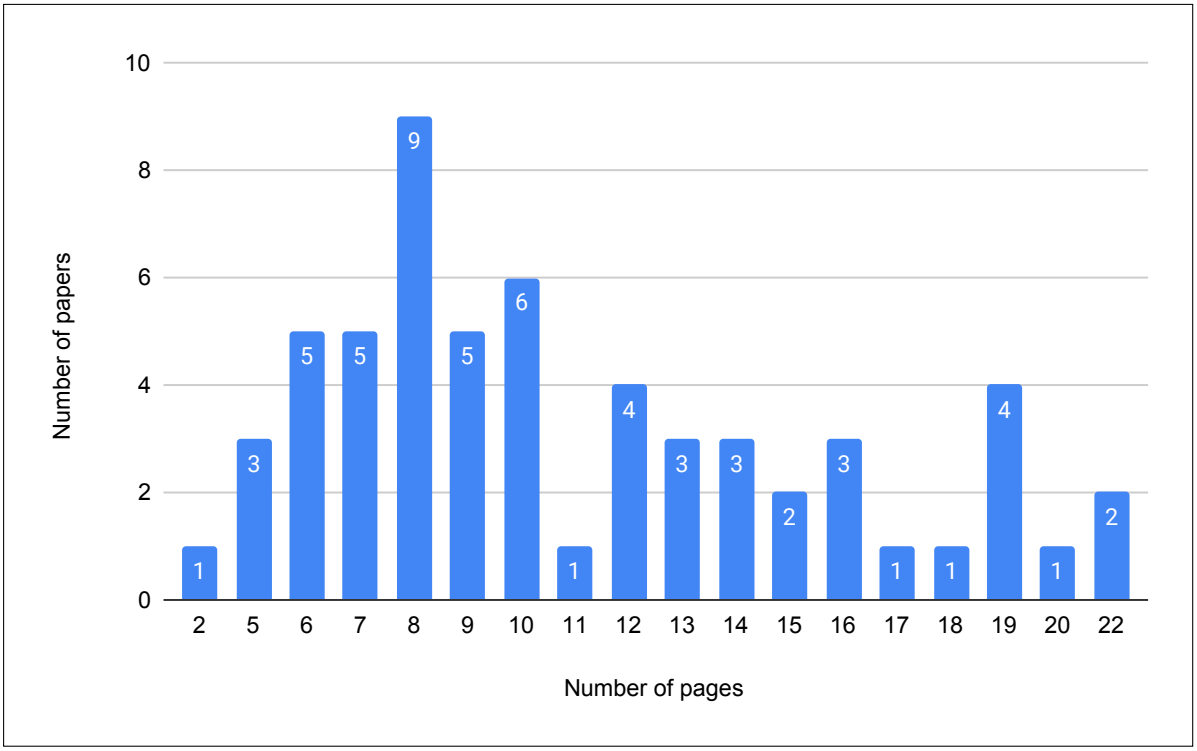


Figure A.4: Quantity of papers by quantity of pages

The percentage distribution of papers by answers (yes, partially and no) for each quality assessment question can be seen in the Table A.4. More than 50 % of the papers present well-organized texts, with motivation and objective well described. Also, more than 50 % of

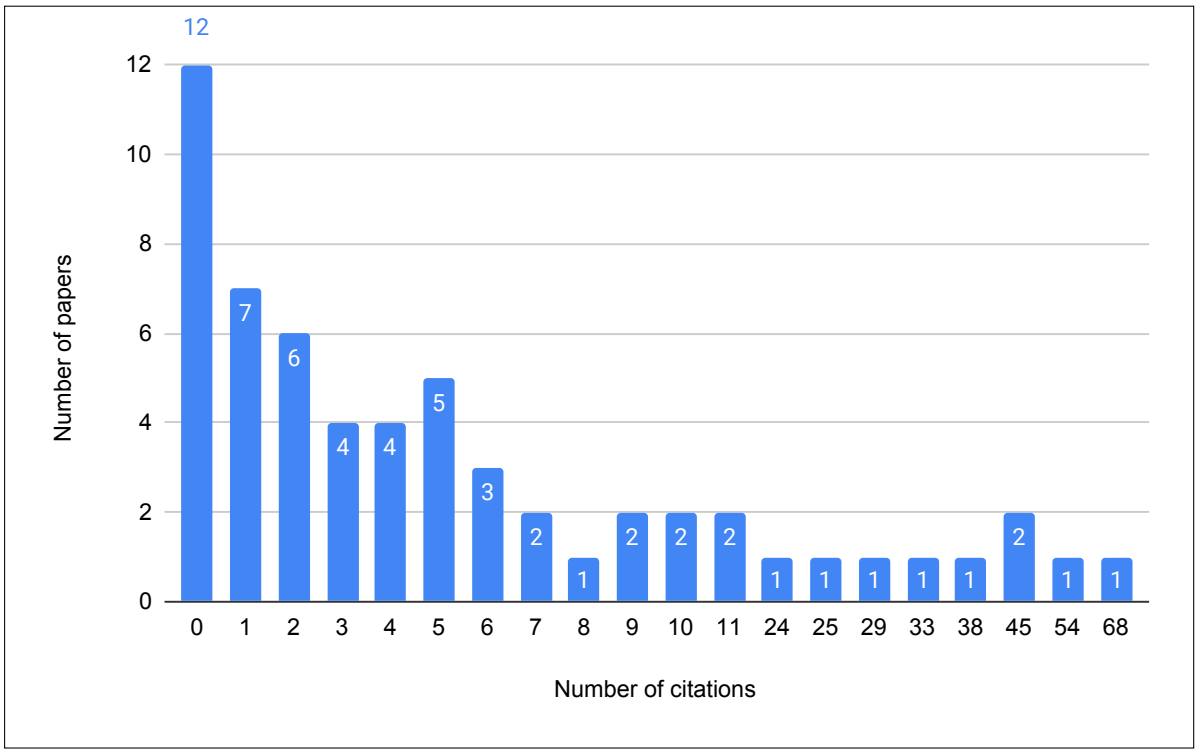


Figure A.5: Quantity of papers by quantity of citations

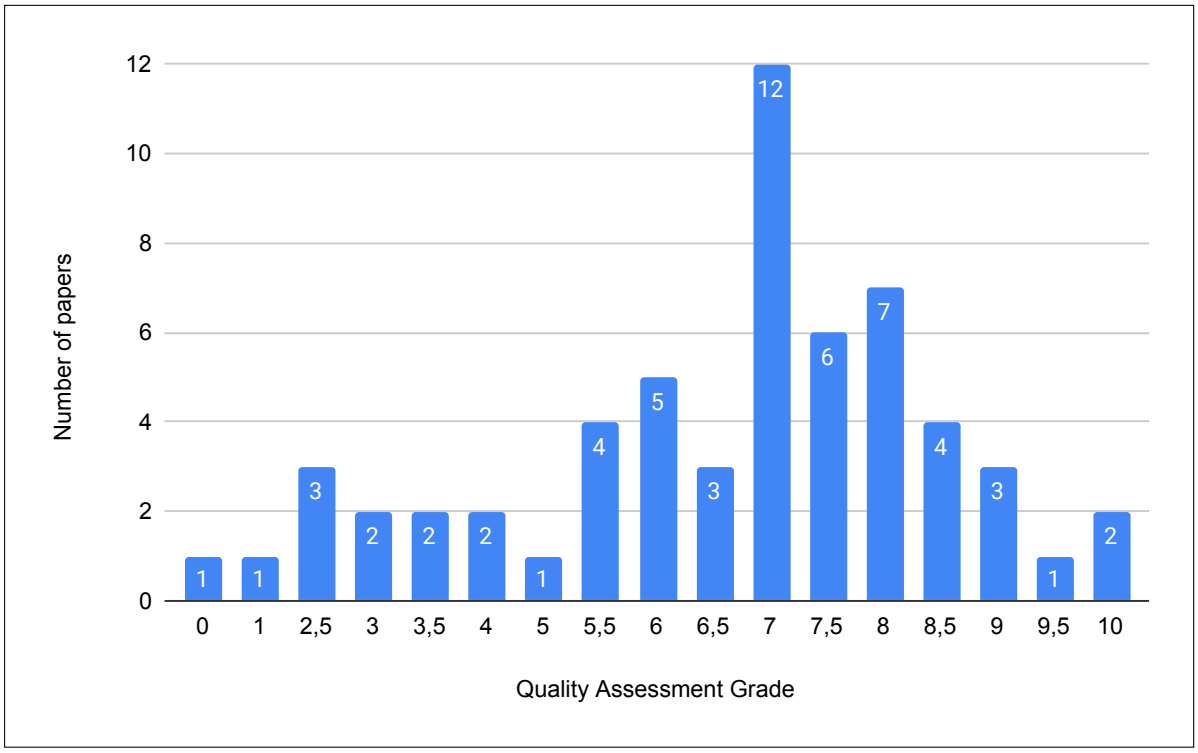


Figure A.6: Quality assessment grades

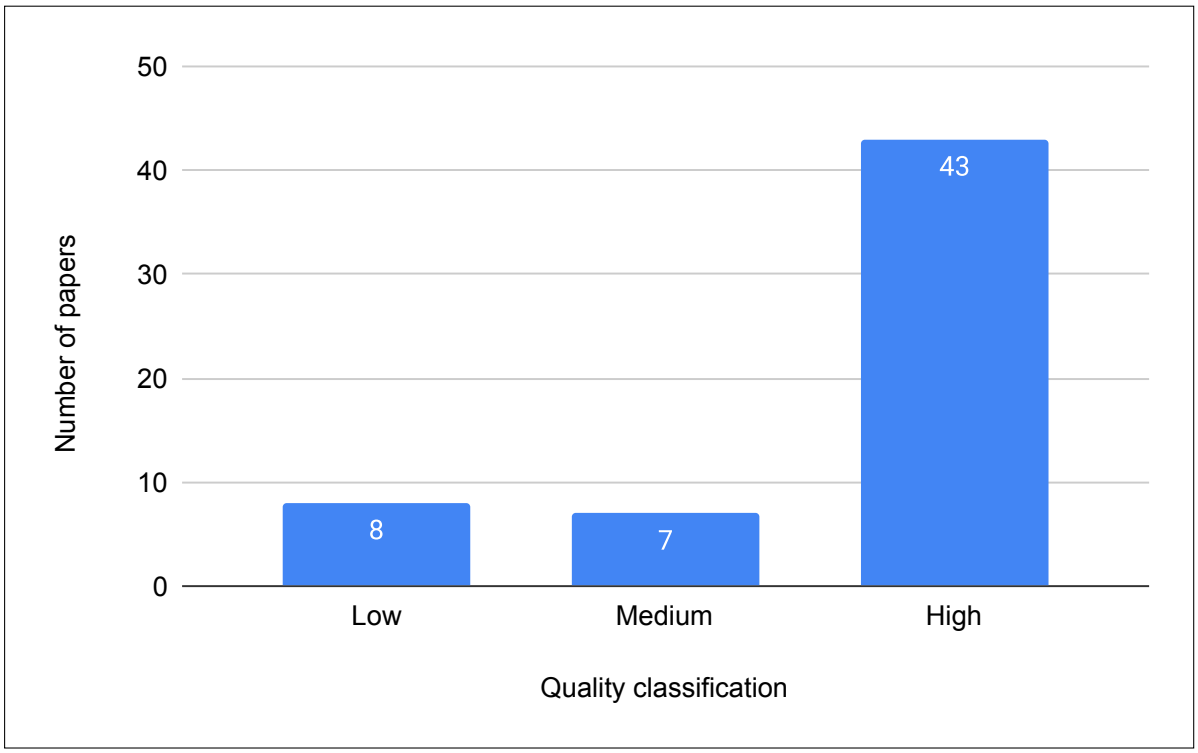


Figure A.7: Quality classification

the papers present many references and good related works and some implementation with practical results. The methodology explanation, the validation, and results discussion are points that can be improved. Lastly, less than 50 % of the papers presented application scenario or case study, clear advantages and disadvantages of the technologies, and suggestions of future works.

Considering the TEE technologies applied in the proposed works, Fig. A.8 presents the proportion of papers applying ARM TrustZone, Intel SGX, or none. "None" was considered when the authors do not mention a specific TEE solution. As seen, most of the papers applied TrustZone technology, which we could expect since it is more suitable for IoT devices due to the ARM architecture. We could also expect that most of the works have applied TrustZone or SGX, as both are the two most known TEE technologies available in the market.

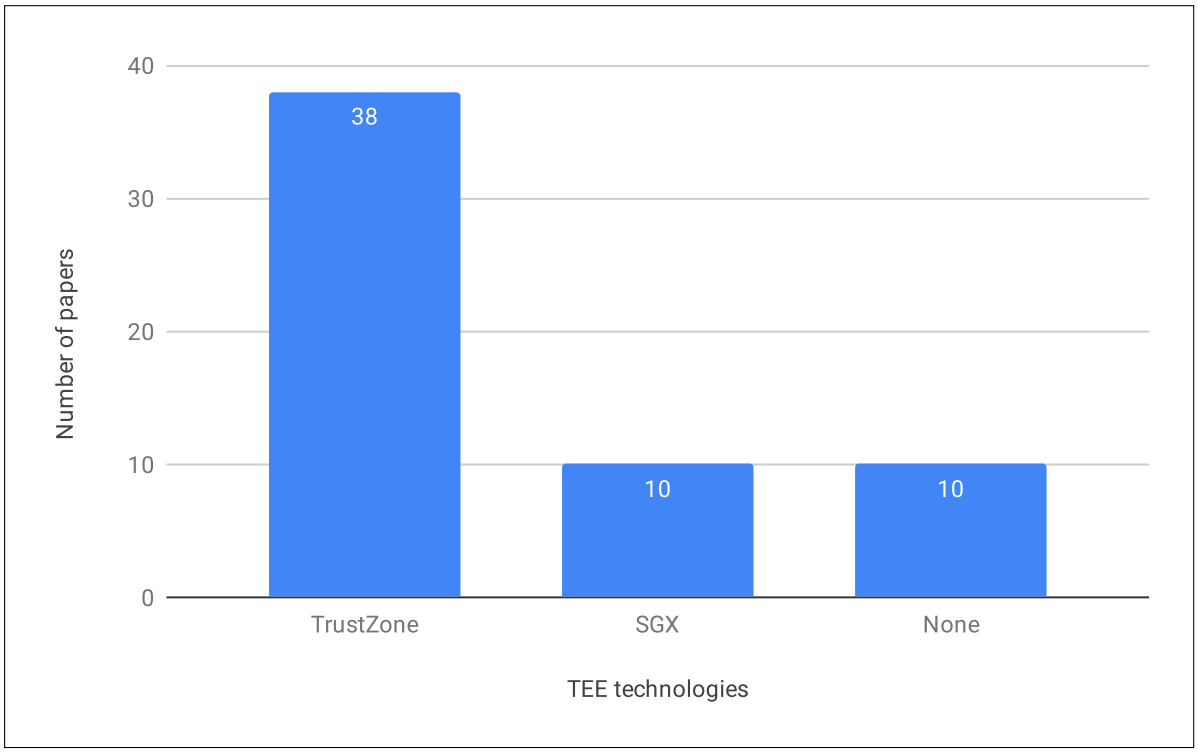


Figure A.8: TEE technologies

Table A.1: Selected papers

Title	Year	Summary
Design and Implementation of Secure Embedded Systems Based on Trustzone [127]	2008	Access control between trusted and untrusted applications
Securing Peer-to-peer Distributions for Mobile Devices [76]	2008	Secure P2P distribution for mobile devices with owner control
Lightweight Anonymous Authentication with TLS and DAA for Embedded Mobile Devices [113]	2011	Anonymous authentication method for mobile devices
Secure device access for automotive software [54]	2013	Secure device access to restrict direct access for the extension software
TEEM: A User-Oriented Trusted Mobile Device for Multi-platform Security Applications [36]	2013	System to address security requirements in mobile and embedded platforms
Hardware Security for Device Authentication in the Smart Grid [78]	2013	Private keys protection without user interaction and device authentication
Trust-E: A Trusted Embedded Operating System Based on the ARM Trustzone [128]	2014	Trusted embedded OS architecture
TrustDump: Reliable Memory Acquisition on Smartphones [102]	2014	Tool to obtain RAM and CPU registers from a compromised mobile OS
Hardware Intrinsic Security to Protect Value in the Mobile Market [63]	2014	Implementation of an electronic fingerprint derived from device physical characteristics
Hardware-security technologies for industrial IoT: TrustZone and security controller [64]	2015	Device snapshot authentication system
Secure and Trusted Application Execution on Embedded Devices [69]	2015	Holistic approach to the security and trust of embedded devices
A Novel Method of APK-Based Automated Execution and Traversal with a Trusted Execution Environment [20]	2016	Tool for lightweight applications using TEE
CacheKit: Evading Memory Introspection Using Cache Incoherence [130]	2016	Rootkit to include malicious code in TrustZone
IoTEE-An integrated framework for rapid trusted IOT application development [124]	2016	Framework for development of trusted IoT applications
C-FLAT: Control-Flow Attestation for Embedded Systems Software [1]	2016	Attestation for the application execution path without requiring the source code
Remote Attestation for Embedded Systems [58]	2016	Implementation of Trusted Platform Module (TPM) with ARM processor
CaSE: Cache-Assisted Secure Execution on ARM Processors [131]	2016	Cache-assisted protection against multi-vector attacks and memory disclosure
A trustenclave-based architecture for ensuring run-time security in embedded terminals [19]	2017	Architecture to secure embedded terminals
TruApp: A TrustZone-based authenticity detection service for mobile apps [34]	2017	Authenticity and integrity verifier for mobile apps
OPTZ: a Hardware Isolation Architecture of Multi-Tasks Based on TrustZone Support [33]	2017	Multitask hardware isolation between normal and secure worlds
TM-Coin: Trustworthy management of TCB measurements in IoT [77]	2017	Trustworthy management system for TCB measurements in IoT
IIoTEED: An Enhanced, Trusted Execution Environment for Industrial IoT Edge Devices [81]	2017	Real Time Operating System inside the secure world of ARM TrustZone
Establishing Mutually Trusted Channels for Remote Sensing Devices with Trusted Execution Environments [96]	2017	Remote attestation in a single run for small-scale devices
MIPE: A Practical Memory Integrity Protection Method in a Trusted Execution Environment [21]	2017	Protection against kernel data and direct memory access attacks
Towards the Security of Motion Detection-based Video Surveillance on IoT Devices [37]	2017	Framework to secure the motion detection procedures
Ditio: Trustworthy Auditing of Sensor Activities in Mobile & IoT Devices [72]	2017	Sensor activities auditing and compliance checking
TrustShadow: Secure Execution of Unmodified Applications with ARM TrustZone [42]	2017	Protection for legacy applications running in untrusted OS
T2Droid: A trustzone-based dynamic analyser for android applications [125]	2017	Dynamic API function calls and kernel syscalls analyser for Android
A private user data protection mechanism in trustzone architecture based on identity authentication [134]	2017	Identity certification of Client Applications to avoid data leakage
TZ-KPM:Kernel Protection Mechanism on Embedded Devices on Hardware-Assisted Isolated Environment [135]	2017	Kernel protection mechanism to detect malicious processes
TICS: Trusted Industry Control System Based on Hardware Security Module [84]	2017	Trustworthiness defense based on attestation and whitelist mechanisms
Boot Attestation: Secure Remote Reporting with Off-The-Shelf IoT Sensors [92]	2017	Light scheme to measure software integrity during the boot phase
Secure mobile device structure for trust IoT [62]	2017	Secure software domain separation to protect IoT data
Cryptographic key protection against FROST for mobile devices [133]	2017	Cryptographic key protection scheme for mobile devices
Encrypting data to pervasive contexts [122]	2017	Communication security between building blocks
LTZVisor: TrustZone is the key [82]	2017	Hypervisor architecture to assist virtualization
An Effective Authentication for Client Application Using ARM TrustZone [53]	2017	Client authentication scheme using ARM TrustZone
ARMHEx: A hardware extension for DIFT on ARM-based SoCs [114]	2017	Protection for DRIFT (Dynamic Information Flow Tracking)
Privacy-Preserving Location-Based Services by Using Intel SGX [56]	2017	User privacy enforcement with anonymity and indistinguishability
Secure and Privacy-Aware Data Dissemination for Cloud-Based Applications [91]	2017	Secure data dissemination for cloud-based IoT applications
On the performance of a trustworthy remote entity in comparison to secure multi-party computation [7]	2017	Performance comparison of SGX-based TRE and secure MPC frameworks
Secure edge computing with ARM trust zone [80]	2017	Platform to connect IoT devices and proprietary cloud solutions
LogSafe: Secure and Scalable Data Logger for IoT Devices [74]	2018	SGX-based trusted logger for IoT devices data
Decentralized IoT Data Management Using BlockChain and Trusted Execution Environment [12]	2018	Decentralized system for IoT data management using blockchain and TEE
Program-flow attestation of IoT systems software [2]	2018	Remote attestation process for IoT devices
Enabling Security-Enhanced Attestation With Intel SGX for Remote Terminal and IoT [115]	2018	Enhance security for remote terminals (bring your own device)
Embedded policing and policy enforcement approach for future secure IoT technologies [99]	2018	Activities monitor on the SoC communication bus
TEE Based Session Key Establishment Protocol for Secure Infotainment Systems [61]	2018	Secure communication between a user device and a vehicle telematics
Feasibility of societal model for securing Internet of Things [108]	2018	Societal model for IoT security, where the "stronger" takes care of the "weaker"
Centralized duplicate removal video storage system with privacy preservation in IoT [126]	2018	Privacy-preserving deduplicate video on storage systems
CryptMe: Data Leakage Prevention for Unmodified Programs on ARM Devices [17]	2018	Integration of memory encryption and TrustZone-based memory access controls
PrivacyGuard: Enforcing Private Data Usage with Blockchain and Attested Execution [132]	2018	Framework that integrates blockchain and trusted execution environment
Development of an Embedded Platform for Secure CPS Services [85]	2018	TrustZone-based platform to secure cyber-physical devices and gateways
Towards Decentralized Accountability and Self-sovereignty in Healthcare Systems [66]	2018	Data Management system to protect personal health data
BASTION-SGX: Bluetooth and Architectural Support for Trusted I/O on SGX [79]	2018	Architectural support for bluetooth trusted I/O
Security and privacy aware data aggregation on cloud computing [100]	2018	Architecture for secure data aggregation in cloud-based IoT
EmLog: Tamper-Resistant System Logging for Constrained Devices with TEEs [95]	2018	Tamper-resistant logging system for constrained devices
Building a Trustworthy Execution Environment to Defeat Exploits from both Cyber Space and Physical Space for ARM [41]	2018	System to shield legacy applications on IoT devices

Table A.2: Conference names

Conferences	
Intl. Conf. on Information Reuse and Integration for Data Science	Intl. Conf. on Information Security Theory and Practice
Intl. Conf. on Embedded Software and Systems	Intl. Conf. on Information Security Practice and Experience*
Intl. Conf. on Information Security	Intl. Conf. on Recent Trends in Electronics, Information & Communication Technology
intl. Conf. on Connected Vehicles and Expo	Intl. Conf. on Trust and Trustworthy Computing
Intl. Conf. on Autonomic & Trusted Computing	Intl. Conf. on Computational Intelligence and Security
Intl. Conf. on High Performance Computing and Communications	Intl. Conf. on Ubiquitous Computing and Communications
Intl. Conf. for Information Technology and Communications	Intl. Conf. on Pervasive Computing and Communications
Intl. Conf. on Wireless and Mobile Computing, Networking and Communications	Intl. Conf. on Availability, Reliability and Security
Intl. Conf. on Trust, Security and Privacy in Computing and Communications*	Intl. Conf. on Utility and Cloud Computing
Intl. Conf. on Mobile Systems, Applications, and Services	Intl. Conf. on the Internet of Things
Intl. Conf. on Field Programmable Logic and Applications	Intl. Conf. on Internet of Things, Big Data and Security
Intl. Conf. on Information and Communications Security	Intl. Conf. on Internet-of-Things Design and Implementation
Intl. Workshop on Data Privacy Management	Intl. Workshop on Smart Grid Security
Intl. Workshop on Hardware and Architectural Support for Security and Privacy	Intl. Workshop on Human-centered Sensing, Networking, and Systems
Intl. Workshop on the Security of Industrial Control Systems and Cyber-Physical Systems*	PerCom Workshop On Security, Privacy And Trust In The Internet of Things
Euromicro Conf. on Real-Time Systems	Conf. on Embedded Network Sensor Systems
European Symposium on Research in Computer Security*	Conf. of the IEEE Industrial Electronics Society
European Symposium on Security and Privacy	Conf. on Computer and Communications Security
Intl. Symposium on Cyberspace Safety and Security	Intl. Wireless Communications and Mobile Computing Conf.
Intl. Symposium on Research in Attacks, Intrusions, and Defenses	Information Security Solutions Europe Conf.
Symposium on Security and Privacy	Thematic Workshops of ACM Multimedia
Learning and Technology Conference	Living in the Internet of Things: Cybersecurity of the IoT

Table A.3: Journal names

Journals	
Tsinghua Science and Technology*	Trans. on Dependable and Secure Computing
The Journal of Supercomputing	Trans. on Computer-Aided Design of Integrated Circuits and Systems
Sensors	Cluster Computing*
Internet Computing - ICT for Smart Industries	Journal of Internet Services and Applications
Design Automation for Embedded Systems	

Table A.4: Percentage of papers according to quality assessment questions

Questions/Answers	Yes	Partially	No
Is the text well organized and clear (easy to understand)?	55,93 %	40,68 %	3,39 %
Is the motivation and objective well described?	76,27 %	20,34 %	3,39 %
Does the paper present an application scenario or case study?	42,37 %	47,46 %	10,17 %
Is the methodology clear (easy to understand and replicate)?	32,20 %	55,93 %	11,86 %
Does the paper have many references and good related works?	52,54 %	42,37 %	5,08 %
Does the study present some implementation and practical results?	71,19 %	18,64 %	10,17 %
Does the paper clearly present any advantages/disadvantages regarding the technology used?	30,51 %	54,24 %	15,25 %
Do the authors present good validation?	25,42 %	52,54 %	22,03 %
Are the results clear enough (explicit and well discussed/evaluated)?	37,29 %	47,46 %	15,25 %
Do the authors suggest future works?	28,81 %	15,25 %	55,93 %