Towards Trusted Apps Platforms for Open CPS

Christian Prehofer*, Oliver Horst*, Riccardo Dodi†,
Arjan Geven‡, George Kornaros§, Eleonora Montanari¶, Michele Paolino‖

*fortiss GmbH, Munich, Germany. Email: {prehofer, horst}@fortiss.org
†e-Services for Life and Health, Milano, Italy. Email: dodi.riccardo@hsr.it
‡TTTech Computertechnik AG, Vienna, Austria. Email: arjan.geven@tttech.com
§TEI of Crete, Heraklion, Greece. Email: kornaros@ie.teicrete.gr
¶Energica Motor Company, Modena, Italy. Email: emontanari@energicasuperbike.com
‖Virtual Open Systems, Grenoble, France. Email: m.paolino@virtualopensystems.com

***Abstract*—**

For many cyber-physical systems, there is a strong trend towards open systems, which can be extended during operation by instantly adding functionalities on demand. We discuss this trend in the context of automotive and medical systems. The goal of this paper is to elaborate the research challenges of new platforms for such open systems. A main problem is that such CPS apps shall be able to access and modify safety critical device internals. We present results of the TAPPS (Trusted Apps for open CPS) project, which develops an end-to-end solution for development and deployment of trusted apps. The main approach is to devise different execution environments for highly-trusted CPS apps. We present the architecture approach and its key components, and methods for CPS apps, including tool chain and development support.

***Index Terms*—cyber-physical-systems, real-time systems, trusted apps, architecture, open-source.**

I. Introduction

Cyber-physical systems (CPS) are devices loaded with sensors and actuators that are able to provide a link between the physical and virtual worlds. An example of such a system is a vehicle that is capable of reading information from the road, and utilizes cloud computing to process this data in a way that would provide new services to the driver.

Currently, in many fields of CPS devices, there is a desire to develop an open system whose functionalities can be extended on demand during operation. The Trusted Apps for Open Cyber-Physical Systems (TAPPS) project focuses on providing this extended functionality through apps, as it is already common within mobile and other consumer domains. The primary concern that is presented here, however, is the considerable security issues that follow the implementation of such a system. A recent research work [1] demonstrates that in a dataset of 22500+ Android apps, 26% of the sample apps were identified as malicious. Considering now the sensitive interactions of CPS systems, including security, safety and privacy aspects, we see trust for such devices as a major societal challenge, which goes beyond the current role of computing in society [2].

In this paper, we first describe the benefits and usage scenarios of a trusted apps platform for several application areas in Section II. This is followed by an analysis of the requirements and research challenges for such platforms in Section III. We continue with a detailed description of the TAPPS approach and all of its components in Section IV and conclude the paper in Section V.

II. Open CPS Application Areas

Cyber physical systems from various application areas could benefit from a trusted apps platform. We will examine the significance of this concept for three application domains: the automotive, healthcare, and industrial automation domain.

A. Automotive

As the average consumer continuously adapts to new IT technologies, in particular on mobile and entertainment devices, there is a rising expectation amongst consumers that the automotive industry will follow suit. While the motivation for implementing new features into produced vehicles is certainly there for producers, there are a number of concerns that have to be assessed before they can be implemented. Security and safety are two of the most critical aspects that must be adhered to when developing the system design and architecture of an automotive. Historically, security was achieved through the isolation of subsystems and with proprietary protocols, however, as the number of interconnected systems rises, tools and software are becoming more available to malicious users, causing reverse engineering and brute force attacks to become a prominent threat to the security of an automotive. It has been reported in literature [3] that there must be coordination between the multiple security technologies that are installed within a vehicle to effectively combat these threats.

The implementation of a trusted apps platform into a vehicular system will aim to provide this required standard of security to the vehicle. Currently, there are a range of applications within a vehicle that vary in how critical an impact they have upon its correct operation. Therefore, it should be a requirement that applications that are untrusted and not critical to the vehicle's correct operation should not be able to influence critical applications in a harmful manner. For example, an untrusted application within a car should not be capable of communicating false information to a critical application within a vehicle that will cause the car to misbehave. A trusted

978-1-5090-1152-0/16/$31.00 ©2016 IEEE

apps platform needs to address the aforementioned security concerns and may do so by controlling the communication between applications within the car.

B. Medical

The use of CPSs in the healthcare domain has been increasingly discussed and analysed by the scientific community in recent years. In [4], this is described as "the combination of embedded software controlling the devices, networking capabilities, and complicated physical dynamics that patients bodies exhibit". Individual vertical solutions for specific use cases are predominant in this domain. A multi-purpose device, where apps may be installed for specific diagnosis or treatments, can have a great impact on the acquisition and maintenance costs of these solutions. The TAPPS project uptakes this idea with a novel and innovative Smart Trolley; a motor-powered medical cart with automatic drawers. The trolley is designed mainly for ward drug administration, but also includes several smart devices, e.g., for patient identification and vital signs monitoring. Issues that may arise with this heterogeneous environment include, but are not limited to: safe drug management to cut down administration errors, prevention of undesired database accesses to protect both patients and users sensitive data, and the correct integration of miscellaneous hardware/software components. The benefits of a trusted apps platform implemented on this medical device can be identified as the separation of different "layers" (i.e., execution environments) of user interaction and data management, according to specific functionalities. For instance, opening/closing drawers containing patient-specific drug therapies must be strictly regulated, while, the collection of body temperatures with a smart thermometer might require fewer checks. The main challenge, in this use case, is the design of an appropriate multi-platform architecture which provides separated connectivity (being extern accesses, or interactions among apps belonging to different execution environments) according to the security level that a specific application/functionality requires, independently from number and type of patients, users, and devices involved in the process.

C. Industrial Automation

Factory automation is changing rapidly due to the increasing need for flexible, reconfigurable and robust infrastructure. The importance of CPS in production facilities is ever-increasing, and is one of the most important innovation drivers for the 4th industrial revolution, i.e., Industry 4.0. In order to allow flexible and reconfigurable CPS to operate in production environments, there is a need for a trusted yet open environment in which existing functionality can be maintained up-to-date, and new functionality can be deployed during the system's lifetime, whilst guaranteeing the trusted execution of control systems at all times. The TAPPS project itself focuses on automotive and medical use cases, but the proposed trusted apps platform was developed with the demands of complex highly cross-linked CPSs, like those found in the industrial automation domain.

III. Requirements and Research Challenges

This section discusses the requirements and expectations of users, app developers, and system integrators towards creating a trusted, open apps platform for CPSs, as well as the associated research challenges.

A. Key Requirements of a Trusted Apps Platform

Cyber physical systems typically expect different software components not to be influenced by each other. This is achieved by intensive tests and complex integration processes and checks. This approach, however, is not sufficient for an open trusted apps platform. A successful approach for a trusted apps platform has to implicitly guarantee the integrity, safety, security and real-time requirements of each app, and the overall system itself under all circumstances. Based on this, the key requirements for a trusted apps platform are as follows:

- *App isolation.* The first and most obvious requirement is that apps need to be executed concurrently without any external influence. This means that a spatial and temporal isolation and separation is required. No app should be able to alter the data or execution behavior of other apps, or the system.
- *Access control.* An essential part of an open trusted apps platform for apps with different criticality and trust levels is the access control to certain functions and resources. Especially the communication with critical interfaces, like, e.g., the CAN bus, needs to be restricted, such that only approved apps can utilize it.
- *Real-time support.* Of special importance for CPSs is the temporal behavior of software components (i.e., apps) and the overall system. This behavior needs to be exactly the same regardless of the utilization of the system, or the apps running in parallel.
- *Resource management.* All resources and peripherals need to be shared in a safe and secure manner between different apps. This requires a management or safeguarding of all resource and peripheral accesses through the execution platform, which has to prevent apps from illegitimately blocking or accessing resources and peripherals, as this may cause damage to peripherals or affect the temporal behaviour of apps.

Comparing the above requirements to existing computing platforms, e.g., on the PC and mobile, we also see sophisticated security and isolation architectures [5]. This is reflected in the first of the above requirements. However, the main difference here is that critical CPSs require even stronger protection, which is addressed by the other three requirements. The main challenge here is that the above requirements induce a certain overhead and complexity to fulfill the required management tasks, which makes it more difficult to apply existing techniques to ensure real-time behavior.

B. Rich Developer Support vs Trusted Real-time Apps

Beside the above requirements, a novel apps platform also needs to attract app developers to be successful. This might

include a feature rich developer support, or even the compatibility with existing platforms and tools. This can enable the reuse of tools, apps, and developer know-how. Unfortunately, a trusted apps platform for open CPSs needs to ensure strong isolation and real-time properties for some apps, which is difficult to combine with existing non-real-time operating systems. Furthermore, existing platforms are typically very complex and support many different APIs and thus also expose many security issues [5]. Hence, it can be preferable to use more limited, smaller operating systems which are easier to check regarding security.

Based on these considerations, we see a research challenge in developing an execution platform that can provide different execution environments (EEs) for different types of apps, depending on their specific needs. For trusted apps which lower criticality, we can offer existing operating systems like Linux or Android, enhanced by security mechanisms discussed below. On the other hand, for highly critical apps which may also require real-time support, we can build on a smaller, real-time aware operating system. This may offer less developer support in terms of APIs and other tools, but provides a more direct control on the system and ensures a more timely behavior.

C. Multi-layered Security and Resource Protection

As an implemented trusted apps platform may be operating upon cyber physical systems where a malfunction could cost lives, it is strictly necessary that multiple layers of security against attacks are provided. The technologies that we have identified as imperative installments into a multi-layered defence against malicious attacks upon a CPS are:

- *Hardware Support.* Hardware mechanisms such as ARM TrustZone [6] provide a hardware based, highly secure mechanism to separate different execution environments. Also hardware components like memory management units and specific on-chip networks can be utilized to physically separate and isolate different execution environments. A hardware based security layer by itself, however, may introduce some limitations. The secure execution environment (i.e., the secure world) of the ARM TrustZone technology, for example, does not support hardware virtualization.
- *Virtualization.* A well known technique to provide virtual execution environments, which also provides protection and isolation of resources, is virtualization. Examples include fully virtual run time environments, like Java, or the virtual duplication of existing platforms via a hypervisor. Virtualization is especially appealing if hardware support is available.
- *Communication middleware.* Resource and access control can be achieved through controlling the communication of apps in the operating system or middleware layer. This can be used to isolate apps from one another in a behavioral sense and to enforce a permission system.
- *App Development Toolchain.* It is also a possibility to permit only trusted apps upon the CPS that have been produced by a specific, trusted tool chain. With the help of software fault isolation [7], [8], or full verification [9] such a toolchain can confine the usage of specific APIs, or can also ensure a proper functional behavior regarding the API usage. It may also be preferable to require the developer to check the app by some trusted third-party, which then cryptographically certifies the correct behavior of the app.

Every CPS is different however, and so one of the larger challenges involved in developing multi-layered security for CPSs is the development of an architecture that combines and selectively utilize the aforementioned technologies for different applications or EEs to individually and specifically meet their requirements. Within the next section, we offer a suitable architecture that will aim to tackle this challenge.

IV. The TAPPS Approach

The architecture we propose within the scope of the TAPPS project addresses all necessary layers from hardware over software to an app store ensuring security and full real-time support for the applications. An overview of the software architecture is presented in Figure 1; the network architecture is given in Figure 2. These figures show the three execution environments and the main security features of the connected TAPPS architecture. For ensuring safe execution of CPS apps, we focus on four key features: the Execution Environments and Apps Platform, the Trusted Inter-EE and Inter-App Communication, the Trusted System and Network Architecture, and the Trusted Development / Model-based Toolchain.

The basic concept of trusted apps relies on the combination of two trust chains. Firstly, a trusted boot mechanism exploits the security on the hardware level and ensures correct booting and installation of the layers of the TAPPS architecture, including the hypervisor, execution environments (EEs) and the app installer. Secondly, the installation of apps is done via a trusted app store, based on a trusted toolchain, which can verify and sign the trusted apps if they comply with the required properties. Furthermore, virtualization and the ARM TrustZone technology is used to separate apps from the system layers below. To provide specific runtime environments for the different apps and their requirements, the TAPPS architecture utilizes three different EEs (see Figure 1). A definition and discussion of these EEs is given in the next section.

A. Execution Environments and Apps Platform

The TAPPS project distinguishes three types of applications: untrusted (U-Apps), trusted (T-Apps) and critical (C-Apps) and therefore proposes an architecture with three different execution environments: the Rich Execution Environment (TAPPS REE), the Trusted Execution Environment (TAPPS TEE) and the Critical Execution Environment (TAPPS CEE).

1) Rich Execution Environment: This is security wise, the least critical environment of the project architecture. It is hosted by a feature rich operating system, e.g., Android or Linux, and is isolated by the KVM hypervisor only. The

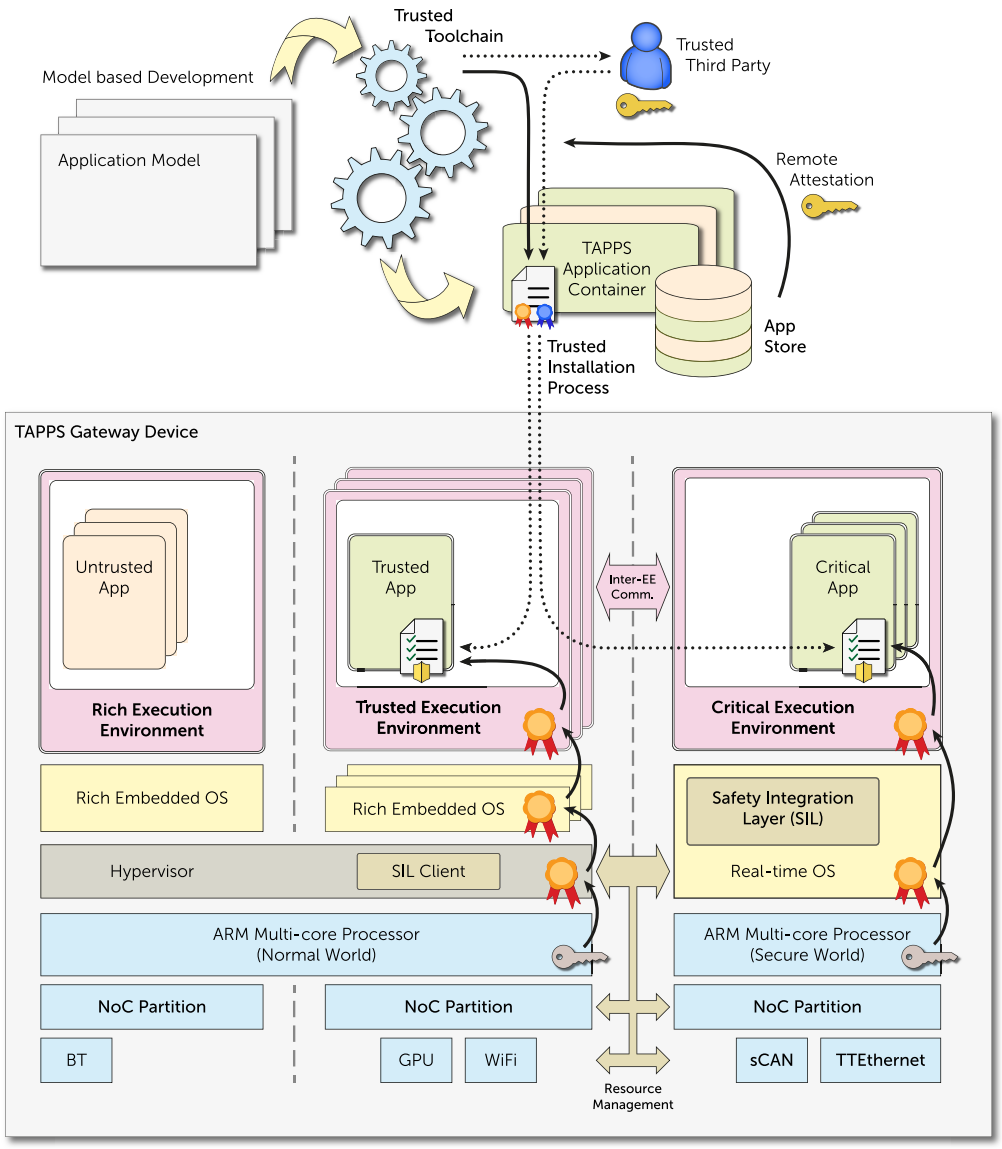

Fig. 1. High-level view on the TAPPS device software architecture and development toolchain.

TAPPS REE will be suitable for hosting untrusted apps (U-Apps), which are similar to those available within the current smartphone ecosystem.

2) Trusted Execution Environment: This is the architecture compartment which runs user applications that interact with the Critical Execution Environment. One or more TEE can be run concurrently in the TAPPS platform, by means of the KVM hypervisor. Each trusted app (T-App) is enclosed into its own hypervisor partition, running within its own virtual machine (VM). T-Apps are the only entities in the system which are allowed to collaborate with critical applications (C-Apps) running in the Critical Execution Environment. This interaction is described in Section IV-B. The TEE is hosted by a hardened, but still feature rich operating system, e.g., Automotive Grade Linux [10].

3) Critical Execution Environment: This is the third, and security/safety wise, most important execution environment of the TAPPS architecture. The TAPPS CEE is designed to run critical soft real-time applications (C-Apps) and is hosted in the Secure World of the ARM TrustZone technology. The operating system that controls this EE shall be small, real-time and safe, like e.g., FreeRTOS [11]. To allow the execution of several individual C-Apps without interference, and to coordinate accesses to peripherals from applications of different trust/criticality levels in the TAPPS CEE, the Safety Integration Layer (SIL) is put in place. This component coordinates the communication between apps, intra and inter EE wise, and supervises accesses to hardware resources. The main objective of the SIL is to guarantee a predictable and repeatable execution of C-Apps including their communication. All applications developed for the TAPPS CEE shall

be developed with the model-driven toolchain, which adds an additional safety layer and the possibility of an objective validation of certain properties with model checking tools.

B. Trusted Inter-EE and Inter-App Communication

The TAPPS architecture defines three EEs able to run different apps which communicate with each other. The trustworthiness of this communication, both between apps running on the same EE (Inter-App) and running on different execution environments (Inter-EE), is of pivotal importance for the security of the entire architecture. For this reason, there is a need to protect, hardware and software-wise, the techniques used by the apps to transmit and receive data/commands to other applications.

To enable Inter-EE communication, TAPPS extends the Global Platform TEE [12] standard, a set of hardware and software directives which aim to implement the concept of Trusted Execution Environment (TEE). The Global Platform TEE is a trustworthy execution environment that stores security sensitive assets and executes trusted functions, in TAPPS this is adopted by the TAPPS CEE. The Global Platform TEE lives together with a Rich Execution Environment (Global Platform REE), the place where user applications are run. In TAPPS this environment is maintained by the hypervisor and shared among the TAPPS TEE and TAPPS REE (cf. Figure 1). The compliance of the hardware platform with this specification is a prerequisite to run the TEE, and it is fulfilled in TAPPS by taking advantage of ARM TrustZone, a set of Security Instruction Set Architecture (ISA) extensions compliant with the standard. The software part of the Global Platform TEE specification, on the other hand, will be adapted and extended during the implementation of the TAPPS Inter-EE communication APIs, in order to fit the TAPPS architecture with its three execution environments.

For what concerns the Inter-App communication, the Safety Integration Layer (SIL) together with a specific network on chip (NoC) implementation, described in Section IV-C, will ensure the authenticity and timing of delivered data.

C. Trusted System and Network Architecture

The multi-layered software architecture, as described in Section IV-A is provided by a tailored system architecture. As shown in Figure 1, the layering is continued in hardware. First of all, we have the in-hardware separation of the normal and the secure world, handled by the ARM TrustZone technology. Secondly, a configurable NoC is in place that can partition the available hardware resources into different protected NoC partitions. With the help of these two concepts, it is possible to assign individual peripherals, or even memory areas to dedicated execution environments with individual access rights.

With the help of specifically trusted communication devices, like a secure CAN (sCAN) and TTEthernet, this offers the possibility to define trusted communication channels, like those discussed in Section IV-B, even through system boundaries.

An elementary component for trusted off-chip communication channels is a secure on-chip communication. In order to achieve this, a novel NoC is integrated into the TAPPS gateway device that partitions the system into several protection domains. The configuration of the NoC is crucial to the overall system safety. Thus, we aim to provide secure programming of the on-chip communication infrastructure by developing a new technology WORMNoC (Write Once, Read Many NoC). This technology allows information to be written a single time, preventing the user from accidentally or intentionally altering it, e.g., protecting the configuration of logical memory compartments or memory-mapped devices. As a result, the WORMNoC will ensure the authenticity of critical data and maintain the on-chip network chain of trust.

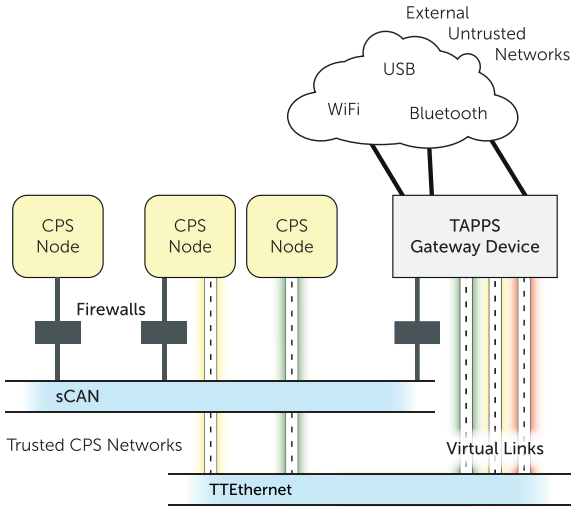

Fig. 2. TAPPS System and Network Architecture.

However, the WORMNoC can only secure the on-chip communication, as soon as the data exits the TAPPS device it has to be safeguarded by a trusted channel. Within the TAPPS project we consider two possible methods, the sCAN bus, and Time Triggered Ethernet (TTEthernet). The objective of ensuring trusted CAN communication is to provide a guarantee to ECUs that safety-critical information transmitted to them has maintained its integrity and authenticity in terms of origin, content, and time. The method that sCAN implements utilises: (1) ECU authentication and (2) stream authorization. In the first step, each ECU authenticates against the TAPPS gateway device. This is performed, for example, when the CPS is not operating and real-time behavior is of limited importance. In the second step, during operation, every message stream is authorized and each CANbus ID between any two nodes that communicate is changing in time on the basis of their private algorithm. Thus, as shown in Figure 2 each CPS node connected to the trusted CAN network is protected by a firewall that manages the aforementioned protection mechanisms. This firewall can either be an external device, or integrated into the CPS node itself, depending on the utilized node hardware. TTEthernet [13] provides strict partitioning of network resources to allow complete segregation of trusted and untrusted traffic in the Ethernet backbone. This partitioning is based on

the concept of encapsulation through virtual links [14]. Traffic streams between applications are mapped to virtual links, which can have different time and safety criticality levels. The TTEthernet network ensures that these criticalities are respected. Virtual links with their criticalities can be mapped to the different execution environments in TAPPS, and as such provide a completely partitioned communication environment for trusted and untrusted traffic.

In addition to securing the CPS internal communication with trusted communication channels, we also envision a network architecture that will ensure the overall safety of the CPS, despite it being open for connections to arbitrary external devices. This network architecture is depicted in Figure 2. Within the proposed architecture we have only one TAPPS gateway device that will adhere to the aforementioned software and hardware architecture. This is the only device that will be allowed to connect and communicate with CPS external untrusted networks, e.g. WiFi, USB, or Bluetooth. This guarantees that all of the TAPPS security layers are in place if someone attacks the CPS through one of the external untrusted channels. Other nodes within the CPS network can only connect through a trusted communication channel, like the aforementioned sCAN or TTEthernet, to each other and the TAPPS gateway device itself. The TAPPS gateway device regulates the trusted network, and thus controls the individual communication between the nodes in the CPS.

D. Trusted Development / Model-based Toolchain

In the TAPPS approach, the toolchain for developing apps is an integral part of the security concept. Only apps developed in a specific toolchain may be permitted into the TAPPS CEE. This is ensured via cryptographically signed application containers that are created by the trusted toolchain, whose trustworthiness is remotely attested by the App Store (cf. Figure 1). Thus, the toolchain is able to limit and analyze the features of an app, even before deployment and on the source level, without having to disclose the source code. For a discussion on tools we refer to [15].

To ensure the greatest possible trust level for the TAPPS CEE, all C-Apps will be developed using a state-machine based modeling framework, e.g., 4DIAC (https://eclipse.org/4diac). This adds an extra layer of security and safety that automatically verifies that developed applications are conforming to the platform API requirements and constraints, ensuring that it will operate safely. Given a finite model of an application and a formal property, it enables us to systematically check whether the property holds for the model, by model checking. This can also be applied to apps, see [16]. For instance, we could check whether an app complies with an agreed communication protocol, or if it shows an intended execution semantic. For an app that controls the daily dosages of medicines for patients, for example, we can check a property to ensure that a patient gets a dosage of medicine if and only if it is prescribed for him. TAPPS will similarly utilise model checking techniques to verify properties of applications.

V. Conclusion

Throughout this paper, we have demonstrated that there is an obvious desire to develop an open CPS, whose functionality can be extended on demand during operation, and have detailed the issues that will be faced when developing such a system. Within such a system, it is essential that apps cannot access and modify safety critical device internals. Our presented architecture, TAPPS (Trusted Apps for Open CPS), and the key concepts that it demonstrates aim to offer an end-to-end solution for the creation of such a system, through the development and deployment of trusted apps, dedicated execution environments for highly-trusted CPS apps, resource protection mechanisms and development support.

It will be a first step for TAPPS to demonstrate a proof of concept with only one TAPPS gateway device per CPS network, however in the future, we see that this proof of concept could be applied to solutions that involve multiple TAPPS devices, which would offer a wider range of possible CPS network configurations.

Acknowledgment

The research leading to these results has received funding from the European Union (EU) Horizon 2020 project TAPPS (Trusted Apps for open CPSs) under RIA grant n° 645119.

References

[1] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *Proc. of the 36th Int. Conf. on Software Eng. – ICSE*. ACM, 2014.

[2] T. Tsiakis, "The role of information security and cryptography in digital democracy," in *Digital Democracy and the Impact of Technology on Governance and Politics*. IGI Global, 2013, ch. 9, pp. 158–174.

[3] I. Security, "Automotive security, best practices," 2015. [Online]. Available: http://www.mcafee.com/de/resources/white-papers/wp-automotive-security.pdf

[4] I. Lee and O. Sokolsky, "Medical cyber physical systems," in *Proc. of the 47th Conf. on Design Automation – DAC*. ACM, 2010, p. 743.

[5] C. Miller, "Mobile attacks and defense," *IEEE Security & Privacy Magazine*, vol. 9, no. 4, pp. 68–70, Jul. 2011.

[6] ARM Ltd., "TrustZone," accessed 12 Feb. 2016. [Online]. Available: http://www.arm.com/products/processors/technologies/trustzone/

[7] R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham, "Efficient software-based fault isolation," *ACM SIGOPS Operating Syst. Review*, vol. 27, no. 5, pp. 203–216, Dec. 1993.

[8] D. Sehr, R. Muth, C. L. Biffle, V. Khimenko, E. Pasko, B. Yee, K. Schimpf, and B. Chen, "Adapting software fault isolation to contemporary cpu architectures," in *19th USENIX Security Symp.*, 2010.

[9] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, P. Schnoebelen, and P. McKenzie, *Systems and Software Verification*. Springer Science + Business Media, 2001.

[10] L. Foundation, "Automotive grade linux," 2016. [Online]. Available: http://www.automotivelinux.org

[11] Real Time Engineers Ltd., "FreeRTOS," 2016. [Online]. Available: http://www.freertos.org

[12] GlobalPlatform, "TEE system architecture v1.0," 2011. [Online]. Available: http://www.globalplatform.org

[13] *Time-Triggered Ethernet*, SAE Std. AS6802, 2011.

[14] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan, "TTEthernet dataflow concept," in *8th IEEE Int. Symp. on Network Computing and Applicat.* IEEE, Jul. 2009.

[15] C. Prehofer and L. Chiarabini, "From internet of things mashups to model-based development," in *IEEE 39th Annu. Comput. Software and Applicat. Conf.*, vol. 3. IEEE, Jul. 2015.

[16] C. Prehofer, "From the internet of things to trusted apps for things," in *IEEE Int. Conf. on Green Computing and Commun. and Internet of Things and Cyber, Physical and Social Computing*. IEEE, Aug. 2013.