IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 April 2023, accepted 1 May 2023, date of publication 15 May 2023, date of current version 30 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3276476

RESEARCH ARTICLE

Mitigating Software Integrity Attacks With Trusted Computing in a Time Distribution Network

DIANA GRATIELA BERBECARU[1], (Member, IEEE), SILVIA SISINNI[1], ANTONIO LIOY[1], BENOIT RAT[2], DAVIDE MARGARIA[3], AND ANDREA VESCO[3]

[1]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy
[2]Seven Solutions (Currently Orolia Spain), 18014 Granada, Spain
[3]LINKS Foundation, 10138 Torino, Italy

Corresponding author: Diana Gratiela Berbecaru (diana.berbecaru@polito.it)

This work was supported by the ROOT (Rolling Out OSNMA for the secure synchronization of Telecom networks) project funded by the European Global Navigation Satellite Systems Agency (now European Union Space Programme Agency – EUSPA) under the European Union's Horizon 2020, the EU Framework Programme for Research and Innovation, under Grant Agreement n. 101004261.

ABSTRACT Time Distribution Networks (TDNs) evolve as new technologies occur to ensure more accurate, reliable, and secure timing information. These networks typically exploit several distributed time servers, organized in a master-slave architecture, that communicate via dedicated timing protocols. From the security perspective, timing data must be protected since its modification or filtering can lead to grave consequences in different time-based contexts, such as health, energy, finance, or transportation. Thus, adequate countermeasures must be employed in all the stages and systems handling timing data from its calculation until it reaches the final users. We consider a TDN offering highly accurate (nanosecond level) time synchronization through specific time unit devices that exploit terrestrial atomic or rubidium clocks and Global Navigation Satellite Systems (GNSS) receivers. Such devices are appealing targets for attackers, who might exploit various attack vectors to compromise their functionality. We individuate three possible software integrity attacks against time devices, and we propose a solution to counter them by exploiting the cryptographic Trusted Platform Module (TPM), defined and supported by the Trusted Computing Group. We used remote attestation software for cloud environments, namely the Keylime framework, to verify (periodically) the software daemons running on the time devices (or their configuration) from a trusted node. Experiments performed on a dedicated testbed set up in the ROOT project with customized time unit devices from Seven Solutions (currently Orolia Spain) allow us to demonstrate that exploiting TPMs and remote attestation in the TDNs is not only helpful but is fundamental for discovering some attacks that would remain otherwise undetected. Our work helps thus TDN operators build more robust, accurate, and secure time synchronization services.

INDEX TERMS Software attacks, time distribution networks, GNSS, trusted computing, remote attestation, keylime.

I. INTRODUCTION

Nowadays, more and more infrastructures - including energy grids, transport, industrial automation services, finance, banking applications, and telecommunication networks - need accurate, reliable, and trusted time for their correct operation. Time synchronization has been traditionally required to support billing, alarm functions, or monitoring of delays in Internet Protocol (IP) networks with accuracy of milliseconds or hundreds of microseconds. In present days, time synchronization is highly required in mobile networks, as well as by applications looking for finer time accuracy, less than 10 microseconds or even less than tens of nanoseconds (ns), such as for MIMO (Multiple-Input Multiple Output)

The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mueen Uddin.

transmitter diversity [1]. Moreover, timing information must be delivered via resilient time distribution networks that can operate and respond in case of malfunctions or deliberate security attacks.

In the telecommunication sector, the availability of accurate timing information has become a fundamental requirement in the Fifth Generation (5G) mobile networks to provide the expected high Quality of Service (QoS). This requirement poses unprecedented challenges for management of multiple timing sources across the 5G networks. In [2], the authors describe possible 5G time synchronization solutions. Some solutions can be implemented in the RAN (Radio Access Network) domain, while others can run in the transport domain. However, a combination of techniques in both domains could be exploited to create a robust and reliable time distribution solution. Since an increasing range of time-based applications are also considered security critical [3], [4], e.g., in the financial, health, transportation, or energy contexts, the timing information needs appropriate protection against cyberattacks. If such attacks occur, they might seriously affect the applications with potentially devastating effects, like the loss of financial information, data compromise, and even human life loss.

This paper describes a solution for mitigating software integrity attacks in a TDN providing time synchronization and distribution in the transport domain. We classify and discuss first the cyberattacks against the installed software (or its configuration) running on the time units operating in an IP-based network. Next, we propose and experiment with a possible solution against software integrity attacks by exploiting a TPM [5] on the dedicated time units supporting the White Rabbit Precision Time Protocol (WR-PTP) [6], [7] for the distribution of ns-level timing information. TPM 2.0 is a cryptographic chip that enables trust in computing platforms. It is widely used to support the creation of trusted environments on servers, laptops, e.g., TPM 2.0 is required to run Windows 11, and in IoT environments [8].

Before explaining the motivation and contribution of our work, we briefly detail the increased need for accuracy and security in the TDNs.

Need for Time Accuracy: The 5G technology offers the benefits of the Time Division Duplex (TDD) [9] spectral efficiency, whose full potential can be exploited if highly accurate time, frequency, and phase synchronization are available at different hierarchical levels of the network architecture. In this sense, 5G networks require increased reliability of the timing sources since a degradation or loss in the timing accuracy and in the precision can impact the RAN performance. On the other hand, legacy networks based on Long-Term Evolution-Frequency Division Duplex (LTE-FDD) can survive lengthy (>1 hour) loss of synchronization without significant performance degradation [10].

The Global Navigation Satellite Systems (GNSS) technologies play a key role in satisfying the stringent requirements for the distribution of an absolute time reference in TDD-based 5G networks: atomic (or sometimes rubidium) clocks may be integrated with GNSS receivers and may be deployed in the TDNs as Centralized Grand Master Clocks (C-GMCs) [1], [11], which generate a Coordinated Universal Time (UTC) traceable time reference. Specifically, the C-GMC obtains a 10MHz clock signal from a rubidium or atomic clock and steers such signal using a One Pulse-per-Second (1-PPS) signal generated by a GNSS receiver. Next, to enable the time distribution to the other nodes in the TDN, the so-called *master* clocks communicate with multiple *slave* clocks through appropriate time protocols. For example, the Network Time Protocol (NTP) [12] allows time synchronization over packet-switched networks. Nevertheless, NTP is suitable for large and dynamic networks requiring time accuracy of a few milliseconds. In case of monitoring of delays in IP networks, time synchronization must be accurate within some hundreds of microseconds. In other applications, the accuracy required is much more stringent, like 5 microseconds for LTE TDD (large cell), 1.5 microseconds for UTRA-TDD, LTE-TDD (small cell), or even hundreds or tens of nanoseconds for cluster-based synchronization [13].

Among the standardization efforts we mention the ITU-T recommendations G.8271 [13] and G.8272 [14] defining time and phase synchronization aspects in telecommunication networks. In time synchronization distributed via packet based methods, the time synchronization is created by a Primary Reference Time Clock (PRTC) [14] exploiting GNSS technology. The G.8272 standard describes a clock that delivers less than 100 ns phase and time performance suitable for packet networks. To increase performance for phase and time required by the emerging mobile access network technologies and improve security for protection against GNSS outages, the ITU-T organization has consented to a new standard (namely, the enhanced Primary Reference Timing Clock - ePRTC) [15]) that called for better performance, i.e., 30 ns, for time, phase, and frequency.

In a network of clocks organized in a master-slave hierarchy, as defined in the IEEE-1588 standard [16], [17], the reference timing signal is carried through dedicated timing protocols, such as PTP (Precision Time Protocol). This protocol can achieve an absolute time synchronization of hundreds of nanoseconds through hardware assistance (SyncE). Furthermore, it can provide sub-nanosecond accuracy through the White Rabbit extension of PTP (WR-PTP).

Need for Security: With the increase of the number of network nodes and to achieve a more scalable solution for time synchronization, the time reference nodes have started to be moved as close as possible to the RAN stations to ''preserve the synchronization budget typically spent across the network hops'' [1]. This trend will lead to multiple atomic clocks and a distributed timing infrastructure based on Distributed Grandmaster Clocks (D-GMCs). On the other hand, having such dedicated time units spread in the transport networks also make them more vulnerable to various types of *security attacks*.

We observe that intentional or unintentional events against time synchronization networks may have a devastating effect. An example of such an attack/event occurred in 2013 [18] affecting Eurex, a famous international stock exchange. In practice, a PTP infrastructure glitch forced Eurex to postpone its market opening because an incorrect leap-second calculation caused an erroneous synchronization of the critical systems. Other incentives the attacker could follow in performing time manipulation are: a) change time backward for movie rentals or digital rights management, b) disrupt forensic analysis process exploiting time for ordering facts, or c) proving to be physically present at a place where he is not [19].

Thus, possible synchronization inaccuracies between the timing sources across the network can directly degrade the performance and the QoS provided by the time synchronization network, leading to reduced throughput, increased latency, or jitter. In extreme cases, a synchronization error can lead to a complete service disruption, which may have a cascading effect on the data managed. For example, in scenarios involving digital identity systems [20], [21], a wrong time reference impacts the application services since, in such systems, the verification of the authentication assertions exploits time. We observe that both the GNSS Radio Frequency (RF) spectrum and the TDN represent viable targets requiring appropriate protection against specific cyberattacks [22], [23].

Motivation and Focus of This Work: The cyberattacks applicable to the GNSS RF spectrum [24], [25], [26], to the PTP protocol [27], [28], [29], [30], or to the WR-PTP protocol [6], [31] have been extensively investigated so far, together with some possible countermeasures. On the other hand, new ideas and solutions are needed to counter the software integrity attacks affecting the dedicated time units in a TDN, though some works have been proposed in this sense (see Section II).

The ROOT (Rolling Out OSNMA for the secure synchronization of Telecom networks) project [32] studied both the attacks against GNSS, like jamming, spoofing, and meaconing (see [1] for detailed description) and the cyberattacks affecting the time units in charge of calculation and distribution of the time reference, that is the D-GMCs and C-GMCs. Furthermore, a selected subset of GNSS and cybersecurity attacks have been tested in a controlled environment hosted at Telefonica's Automation and Innovation Lab (Madrid, Spain). Last but not least, the project evaluated the performance of OSNMA (Open Service Navigation Message Authentication) signal [33] in the GNSS receivers placed on the exploited time units.

This paper deals with the protection against software integrity attacks targeting the nodes operating both as C-GMC and as D-GMC of the ROOT time distribution network. We assume that a trusted operating system runs on these devices. Moreover, the network attacks are not considered, though we underline their relevance in the classification presented in Section II. The ROOT architecture exploited tailored time synchronization devices from Seven Solutions (named WR-Z16 [34]) operating at different TDN layers. As said, these devices generate time reference by combining different time sources (GNSS, cesium atomic clock, rubidium clock, or Oven-Controlled Crystal Oscillator depending on whether they behave as C-GCM or D-GCM), and are capable to grant *sub-nanosecond* synchronization accuracy by exploiting specific software and the WR-PTP protocol.

To protect from software tampering attacks on the dedicated time unit devices, we exploited a TPM, as well as specific Trusted Computing (TC) and remote attestation software. More specifically, we used a software TPM installed on the dedicated time units, along with software implementing the TCG (Trusted Computing Group) software stack specification [35] to interact with the TPM. To measure the software components, like executables, configuration files, and kernel modules, we exploited IMA (Integrity Measurement Architecture) [36], which is part of the Linux kernel since 2009. In addition, we used the Keylime framework [37] to perform TPM-based remote attestation of the time units from a remote trusted node in a highly scalable manner.

We detail the experimental results performed on a real testbed exploiting the WR-Z16 devices installed in a controlled environment, the same one used for GNSS attacks evaluation described in [1]. We show how the Keylime remote attestation software can mitigate a selected set of software integrity attacks against the considered devices. In practice, we have tested the following ones: 1) modification of the software configuration on the time unit. 2) alteration or replacement of the executable software on the time unit. 3) change the GNSS receiver configuration placed on board the WR-Z16 devices.

Organization: The paper is organized as follows. Section II presents shortly the related work addressing cybersecurity attacks and countermeasures in time synchronization networks. Section III introduces the ROOT architecture (at a high level), Section IV gives an overview of the attacks that might affect a TDN and classifies them into categories based on different views, namely hardware, software, time distribution/synchronization, or management. Section V details the main trusted computing concepts and technologies used for software integrity protection on the time units, Section VI describes the experimental testbed software (including the Keylime software) running on the TDN nodes, Section VII illustrates the experiments performed on the installed testbed, Section VIII presents the results obtained in the experimental attacks. Finally, Section IX draws the final remarks and conclusions.

II. RELATED WORK

The series of works of Alghmadi and Schukat [38], [39], [40], [41] is closest to the work reported in this paper since they have addressed both the network attacks as well as the advanced persistent threats [42], [43] and internal attacks [44] targeting the software or the configuration of the time devices. Nevertheless, the abovementioned research papers have not

TABLE 1. Summary of (selected) related papers.

Ref	Year	Short description	TC-enabled experimental TDN?
[38]	2020	Provides a comprehensive analysis of strategies for APTs to PTP infrastructure, possible attacker locations, and the impact on clock and network synchronization.	No
[39]	2021	Proposes and demonstrates a programmable MITM (pMitM) and a programmable injector (pInj) device allowing the implementation of a variety of attacks to quantify the impact of APTs on time synchronisation.	No
[40]	2020	Investigates the impact of the most important APT (Advanced Persistent Threat) strategies on a PTP network, i.e., the delay attack, packet modification or transparent clock attack, and time reference attack.	No
[41]	2022	Proposes a detection system that analyzes data collected from all slave clocks in a TDN during the synchronization process and then monitor for anomalies caused by an attack.	No
[44]	2017	Discusses how countermeasures based on public key infrastructures, TPMs, network intrusion detection systems and time synchronization supervisors can be adopted to defeat or detect internal attacks in a TDN.	No
[19]	2019	Discusses the exploitation of TEEs (like Intel SGX, TPM, ARM TrustZone) for securing time.	No
[45]	2021	Introduces a preliminary classification of cybersecurity attacks affecting TDNs and proposes a TC-based solution to counter software integrity attacks.	No
[46]	2021	Analyzes the principal attack vectors in Industry 4.0 applications and the need for software integrity controls for TDN nodes. Builds a testbed prototype with TC-enabled IoT devices achieving microsecond synchronization accuracy.	Yes

exploited TC or remote attestation experimentally in a TDN, as presented in this work. For example, [44] discusses how countermeasures based on public key infrastructures, trusted platform modules, network intrusion detection systems and time synchronization supervisors could be adopted to defeat or at least detect such internal attacks but still lacks practical results on employing such technologies in a time synchronization network. The importance of trusted time for various applications is discussed in [19]. The authors examined in a detailed manner the timing capabilities of various TEE (Trusted Execution Envirnnments) that put their root of trust in hardware, including Intel SGX, ARM TrustZone, and TPM. They established that these technologies are susceptible to timing attacks by a malicious operating system and an untrusted network. The same paper ([19]) argued that it is essential to protect time-based primitives across all layers, i.e., the hardware timers, platform software, and network time packets, even if it did not perform an experimental evaluation in this sense.

Some authors of the current paper sketched a preliminary classification of attacks in TDNs and designed a solution for TC-based time distribution in an environment using a Raspberry Pi 4 device (used as an attester node), a trusted Verifier node, and the Keylime software [45]. In parallel, the authors in [46] have analyzed the principal attack vectors in the context of Industry 4.0 applications and highlighted the need for software integrity controls of the TDN nodes. They have also proposed a solution based on TC principles to protect the integrity of the nodes of a time synchronization network. Furthermore, they demonstrated the effectiveness of the proposed approach with a simplified testbed based on low-cost Internet of Things (IoT) devices capable of achieving microsecond-level synchronization accuracy.

In the past, the security of the time information in transit over the TDNs has been extensively studied, mainly because unsecured formats were employed for transporting such data [47]. Some works, such as [48], adopted secure data formats, whereas other authors [49] proposed secure timing protocols, like NTP secured through the Network Time Security (NTS) protocol. To secure broadcast time synchronization, [50] set forth using data origin authentication.

Regarding remote attestation, several possible implementations exist. Several works performed remote attestation by experimenting with different hardware or software solutions [51], [52], [53]. Some solutions used additional hardware features to perform the attestation, like Intel TXT [54]. A survey of (hardware) remote attestation schemes for various contexts, including cloud computing, IoTs, and critical network infrastructures, can be found in [55]. Others (mainly for embedded systems) removed the hardware components suggesting a solution relying entirely on software [51]. In the IoT networks, the current trend goes toward collective remote attestation (CRA) schemes [56] capable of remotely performing attestation of large networks of (IoT) devices. Selected papers tighlty related to the topic of this work are resumed in Table 1.

III. THE ROOT ARCHITECTURE: HIGH-LEVEL VIEW

To achieve high-accuracy time distribution (e.g., from 65 ns to 130 ns for 5G front-haul applications) and secure time synchronization, the ROOT project designed and experimented with a network architecture exploiting centralized master clocks and distributed grandmaster clocks.

A high-level view of the ROOT architecture is shown on the right hand in Fig. 1, while the left side of the figure shows internal details of the C-GMC, D-CGMC, and Boundary Clock(BC)/Grandmaster(GM). A C-GMC generates a time reference by combining different time sources. In particular, the C-GMC uses a GNSS receiver together with a co-located Cesium Atomic Clock (Cs AC), as shown in Fig. 1 a). Multiple distributed reference clocks, namely the D-GMCs, are spread across the network to provide robustness and

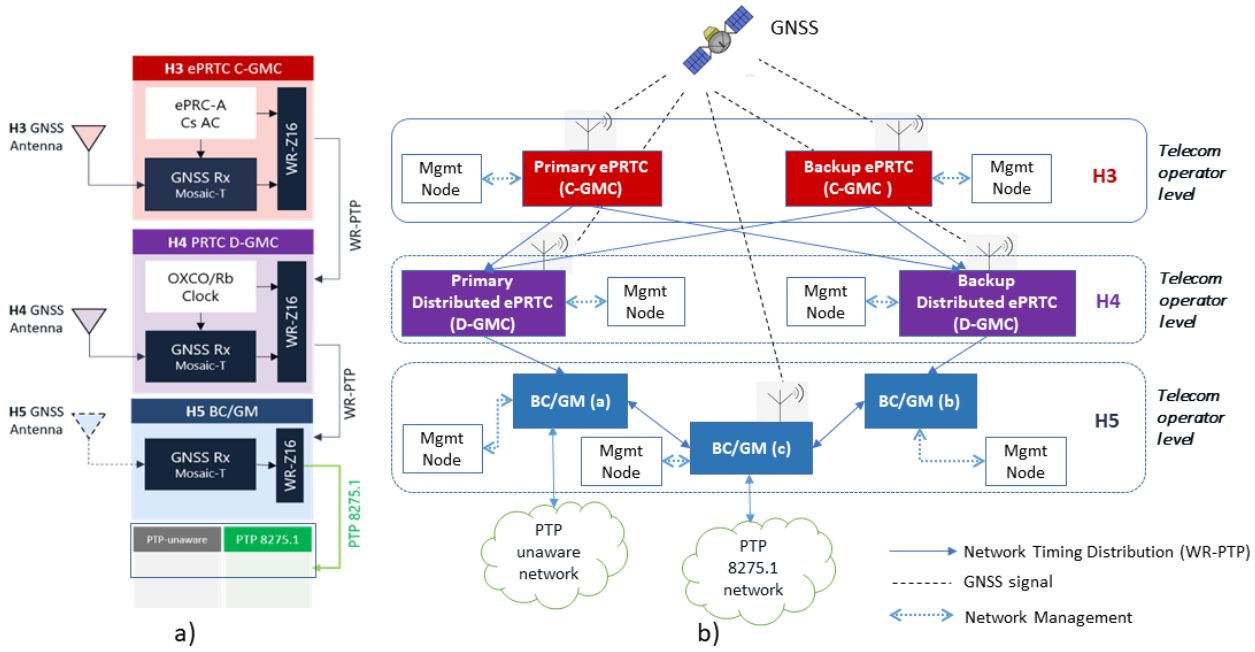

FIGURE 1. ROOT time synchronization architecture exploiting WR-Z16 devices acting as C-CGM and D-CGM deployed at different levels of the telecom network operator.

resilience in case of temporary malfunctioning or intentional attacks. Specifically, the D-GMC combines multiple time sources to generate a time reference, that is, a GNSS receiver and a (less expensive) Oven-Controlled Crystal Oscillator (OCXO) or a Rubidium (Rb) clock. For resilience purposes, the proposed TDN exploits two C-GMCs and two D-GMCs, namely a primary and a secondary one. Generally, the primary and secondary devices have similar configurations, but they operate in different physical locations. The time synchronization signal is transported to the connected devices over fiber optical channels by exploiting the WR-PTP protocol.

The telecom network operator installs the C-GMC and D-GMC devices at different operational levels. In the TDN architecture shown in Fig. 1 b), the C-GMC and D-GMC devices occur at the regional level (H3) and the metro aggregation level (H4) of Telefonica telecom operator network (more details on the so-called FUSION hierarchy levels can be found in [1]). At the hierarchical level (H5) operates a less complex time distribution device (called BC/GM), which exploits as well a GNSS receiver for time calculation and no terrestrial clocks. The H5 level is the most distributed one where mobile base stations connect.

IV. CYBERATTACKS APPLICABLE TO THE TDNs

In [45], we preliminary classified the attacks affecting TDNs, by considering different possible attack vectors. Concerning the TDN architectures, an adversary may pick one or a combination of the following security attack scenarios:

- GNSS attacks against the communication links between the satellites and the GNSS-enabled nodes in the TDN. An attacker might calculate and broadcast a fake GNSS signal (a.k.a. spoofing attack [57], [58]), retransmit a real but delayed GNSS signal (meaconing attack), or launch denial of service attacks (including jamming [59]); These attacks affect the GNSS signal links shown in Fig. 1 b).
- hardware attacks - deal with physical corruption of the specific devices. Examples include heating, cooling, or physical manipulation of parts of the device to deviate from its ''normal'' operation. We do not consider these attack types in this work because we assume the C-GMC and D-GMC devices are resilient to such attacks or operate in protected locations.
- time distribution attacks - regard all the attacks discovered so far against the time distribution protocols (i.e., PTP or WR-PTP) such as: denial of service (DoS) attacks, man in the middle (MITM) attacks, replay attacks, or delay attacks [31], [60]. These attacks affect the network timing distribution links shown in Fig. 1 b).
- network attacks - refer to the networking attacks, including DoS attacks, MITM attacks, and replay attacks, targeting the network management protocols employed by a legitimate network administrator to perform remote access (and authentication) to the time device(s) for management and configuration purposes. Examples of such protocols are the Secure Shell (SSH), Transport Layer Security (TLS) [62], or SNMP (Simple Network Management Protocol). These attacks, such as MITM SSL [63], or TLS truncation attack [64], [65] affect the network management links shown in Fig. 1 b). They

can be damaging because, if successful, the attacker can actively intercept authentication credentials (allowing him/her to connect as a legitimate user). Moreover, the attacker could alter the installation code or configuration parameters sent to the time devices.

- software attacks against the specific D-GMC or C-GMC devices, including:
  - modification of the specific software or its configuration through unauthorized access to the device (including software engineering attacks), installation of malware, or software backdoors. In this category are included also the memory scraping and the side-channel attacks, as well as the exploitation of buffer overflows in the installed operating system or software;
  - installation of malicious code/software, either through injection attacks [66], worms/trojans, ransomware, rootkits, botnets, or malicious network functions.

The architecture shown in Fig. 1 b) can thus be further decomposed into different views, namely *time distribution*, *network* (management), *hardware*, and *software*. For each view, multiple potential attacks exist. Theoretical or practical attacks could exploit vulnerabilities in the timing protocols, the network management protocols, the physical protection, or the software installed (and configured) on the time unit devices.

In this paper, we concentrate on the software view. Thus, we consider the entire software stack running on the time unit devices, including the operating system, the management software, and the specialized daemons employed for time synchronization, such as `ptpd` [67], `gpsd` [68], or `ppsi` [69]. In general, software attacks - sometimes called low-level attacks - rely on characteristics of the hardware, compiler, or operating system used to execute software programs to make these programs misbehave or extract sensitive information from them. A possible solution to detect intentional changes in the software running on the C-GMC or D-GMC devices could exploit trusted computing by taking advantage of a TPM to support trusted or secure boot, the protected storage of the TPM for sensitive data, and to perform remote attestation [70].

V. PROTECTING FROM SOFTWARE INTEGRITY ATTACKS WITH TRUSTED COMPUTING

This section provides the background information necessary to understand the TC techniques used to counter the software (integrity) attacks in TDN devices. Then, we present in brief the Keylime framework.

A. TPM 2.0, CHAIN OF TRUST, TPM KEY HIERARCHIES

The Trusted Computing Platform Alliance (TCPA), formed by companies like Microsoft, Intel, IBM, Hewlett Packard (HP), and Compaq started to promote in 1999 the development of Trusted Computing relying on both hardware and software implementations. The TCPA proposed a hardware anchor for PC security, namely the TPM, which can be used as a base to build secure systems. The core idea behind this chip stands in protecting (efficiently) the device's cryptographic assets [71]. The TPM is now present in almost all commercial PCs and servers, with specifications managed by the TCG (Trusted Computing Group) (successor of TPCA), whose members are famous companies like AMD, Hewlett Packard, IBM, Intel, Microsoft, Sony, or Sun Microsystems. Several forms of TPM exist nowadays [72]: the *discrete* and *integrated* hardware TPMs are used in critical systems or gateways and offer a higher security level and have high-to-medium costs, the *virtual* TPM (typically used in cloud environments) is a software implementation of the TPM running in an isolated execution environment and has a low cost, whereas the *software* TPM runs as a regular program within the operating system. The software TPM does not offer any security guarantee because it is subject to physical tampering and (software) attacks addressing the operating system on which it runs and the software TPM itself. However, it is appropriate for testing (purposes) since it allows one to emulate an environment with TPM-enabled applications and measure its performance. We have used such a TPM in the solution described herein.

The TPM must be secure itself. It can be seen as a Root of Trust providing confidentiality, protection, and integrity services even if it's not robust against hardware-based attacks. A key aspect of the TPM is the *identification*, which means it can manage identity keys, more precisely, an endorsement key (EK) pair representing TPM identity. According to the TCG specification, a platform is *trusted* if *it behaves as expected* for a *specific purpose*. Thus, it's imperative to determine the identity of its hardware and software components. The TPM (as proposed by TCG) allows to collect and report these identities in a way that permits to determine the expected behavior and (from that expectation) to establish trust [73]. The TPM is separate from the host system, interacting with it only through the interface defined in the TCG specification. The TPM specification 2.0 published in 2014 [74] supports many more new features and algorithm agility in contrast to the previous version (namely 1.2).

Relevant to this work are the key hierarchies and the Platform Configuration Registers (PCRs) in the TPM. The PCRs are shielded locations of memory used to validate the measurements made by the system to its internal components. These registers hold measurements of the software done through a hash function, starting from the BIOS (loaded at device boot) up to regular files or configuration files accessed at application run time. PCRs 0 to 9 store measurements about BIOS, firmware, and boot loader, while PCR 10 stores measurements about applications, made with the Integrity Measurement Architecture - IMA. The content of the PCRs can be modified only at reboot time, or with a particular operation called *extension*. The PCR values are cleared every time the TPM is restarted, usually, the values are set to all zeros, and they cannot be erased or modified manually. As said, the

only operation that can be done on a PCR is the extension, through an operation indicated below, where $PCR_{newvalue}$ is the new hash values stored in the PCR after the *extension*, $Hash_{hAlg}$ is the hash algorithm used for hash calculation associated with a specific PCR bank, $PCR_{oldvalue}$ is the PCR value before the *extension* operation, and $data - hash$ is the new data (measurement) that must be protected:

$$PCR_{newvalue} = Hash_{hAlg}(PCR_{oldvalue}||data - hash)$$

The PCRs'content can be read internally or externally to validate the state of the system, usually through a TPM *quote* operation. Besides the cryptography subsystem that implements all TPM's cryptographic functions (such as the hash engine, the asymmetric cryptographic engine, the symmetric encryption engine, and the random number generator used to generate keys securely), the TPM has additional features ensuring the security of the node on which is hosted. The TPM allows, for instance, the secure generation of keys (via a random number generator) and the secure storage of objects (including keys) that are accessible only through protected capabilities preventing thus unauthorized deletion, modification or disclosure.

1) ROOTS OF TRUST

The core idea in trusted computing is the creation of a *Chain of Trust* starting from a Root of Trust (RoT), a component considered trusted since its misbehavior is not detectable. For example, a certificate provided by an independent testing lab may report the Evaluation Assurance Level of the TPM, providing thus confidence in the correct implementation of its RoTs. A Root of Trust is the minimum set of system elements that convey trustworthiness to a platform and is strongly bounded to the concept of a Trusted Building Block. Since the characteristics affecting the trustworthiness of a platform are multiple, a more appropriate term is *roots of trust*, where each one of them is the starting point for one of the parts in which the process of trust attestation is divided. According to its specification, the TCG group requires that a trusted platform provides (at minimum) the following three roots of trust: the Root of Trust for Measurement (RTM), the Root of Trust for Storage (RTS), and the Root of Trust for Reporting (RTR).

The RTS is the TPM memory, which has the property of being shielded from access by entities other than the TPM. The RTM supports the integrity measurement of the TPM by calculating digests taken on the configuration data and program code and sending them to the RTS. The integrity measurement of a platform exploits the *transitive trust* concept, meaning that the trust in a software component is used to evaluate the trustworthiness of the subsequent software component that will take control of the platform.

The RTM of a platform is typically a small subset of the BIOS, called Core Root of Trust for Measurement (CRTM), which the CPU executes at system reset. The CRTM contains the first set of instructions that get control of the system. Its purpose is to record in the TPM which BIOS is used to boot the system before passing control to the full BIOS. The CRTM is the starting point of a chain of trust, transferred to subsequent software components.

The RTR reports on the contents of the RTS. Typically, an RTR report is a digitally signed digest calculated on the values of some shielded locations within a TPM, such as the content of PCRs, which provide evidence of the platform status. In this case, the RTR report is called Integrity Report (IR).

2) TPM KEY HIERARCHIES

The TPM reports on the integrity state of a trusted platform by quoting the PCR values. Yet, external entities may check the platform state through quote operation provided that the TPM issuing the quote (actually, the RTS inside it) identifies itself. The RTR (and TPM) identification is accomplished through non-migratable asymmetric keys called Endorsement Keys (EKs), derived from an endorsement *seed* contained in the TPM. The seed (large random number that never leaves the TPM) is probably the most significant element. It acts as a starting point from which primary keys are generated. Subsequently, from these primary keys, other objects are derived, namely the proof values used by the TPM to attest its identity when sending some data. The TPM supports three types of persistent hierarchies: a) the *platform hierarchy*, used to ensure the integrity of the system firmware; b) the *storage hierarchy*, used by the platform owner for a variety of non-privacy-sensitive purposes; c) the *endorsement hierarchy*, used when the user wants to ensure the integrity of privacy-sensitive data, attesting TPM identity since its primary keys are guaranteed to be constrained and unique to each TPM by the manufacturer. The keys of this hierarchy are also known as Endorsement Keys (EKs) and are created starting from the unique endorsement seed contained in the TPM. These keys are also used in critical tasks, including the remote attestation process.

The endorsement hierarchy and the EK have a key role in the platform identification: the RTR and the reporting phase itself are meaningless if not strictly linked to the TPM they refer to. In other words, a proof of the physical bounding between the RTM and the RTR is needed to be sure that a particular quote was sent by a well-defined TPM. This property is achieved through the EK since it is almost impossible to have two TPMs with the same endorsement seed, that is, with identical endorsement keys. The EKs can raise privacy problems: since they represent the TPM identity, their direct use could permit the creation of activity logs, which could reveal personal information that the user of a platform would not otherwise want to reveal to entities that aggregate data [73]. To counter this issue, TCG recommends not to use EKs either for data signing or encryption operations, but only to decrypt certificates of other non-migratable keys generated by the TPM, namely the Attestation Keys (AKs) or Attestation Identity Keys (AIKs). These keys are obtained during the Attestation Key Identity Certification procedure, which attests that an AK has been generated by an authentic

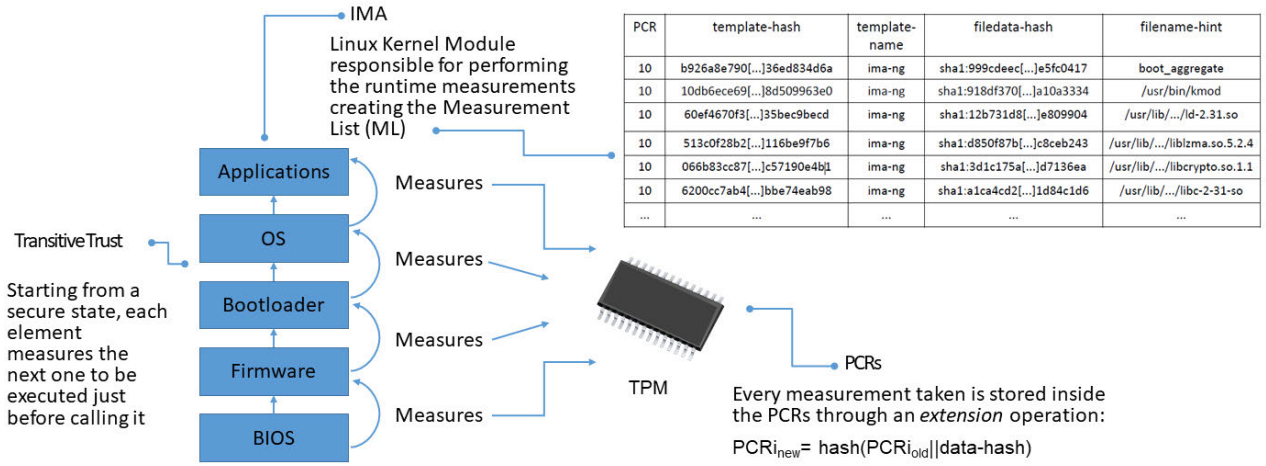

FIGURE 2. Software integrity measurement components (exploiting IMA and TPM).

but unidentified TPM, is a non-migratable key, and can be used to sign the contents of shielded locations. This happens upon request of the TPM owner with the cooperation of an Attestation CA (or Privacy CA). Overall, its purpose is to certify that a given AK has been generated by a valid TPM. AKs, in turn, can only be used to sign digest values that the TPM generates, like the digest calculated on the content of PCRs.

3) MEASURING SYSTEM INTEGRITY WITH IMA

The TCG specifications define all the concepts staying at the base of TC philosophy, including the definition of the RoTs that a trusted platform has to implement. Moreover, TCG specifications define the building blocks for creating a Trusted Boot. In particular, the CRTM and the transitive trust concept allow for establishing an RTM, which measures the system boot components and stores the measurements in the RTS, more specifically, into the PCRs in the TPM. Thus, an external entity may verify that the system booted securely. However, TCG specifications are operating system (OS) agnostic. They do not state how the RTM of the platform extends in the OS. The Integrity Measurement Architecture (IMA) is the Linux kernel's implementation of the integrity measurement system conceived by the TCG. It allows extending the chain of trust from the BIOS up to the application layer, as shown in Fig. 2.

IMA is currently one of the most accepted TCG-compliant solutions for measuring dynamic executable contents [75], being part of the Linux Integrity Subsystem starting with the version 2.6.30 of 2009. IMA measures all the executables, configuration files and kernel modules as soon as they are loaded onto the Linux system before passing the control to them. Then, it extends these measures in the TPM, more precisely in the PCR 10.

Yet, IMA does not measure everything, but it processes all files accessed at runtime according to some rules specified in the *IMA policy* configured on the system. The policy can be one of the IMA built-in policies allowing to measure the Trusting Computing Base of the system, or it can be a custom policy created according to the needs of a specific context. The files are measured by means of a digest applied over their complete content, using the SHA1 algorithm (by default) or a hash algorithm specified through a kernel command parameter, like SHA256. All the measurements are stored in a Measurement List (ML) inside inside two log files in the security file system (named `ascii_runtime_measurements` and `binary_runtime_measurements`) so the measurements are available in the user space. IMA allows to store in the ML other metadata along with the file measurement by exploiting a set of built-in templates, such as `ima`, `ima-ng` (default), or `ima-sig`. When a new entry is added to the ML, IMA computes a digest on all fields specified by the IMA template, and if the platform is equipped with a TPM, it extends the digest in the PCR 10.This process permits an external entity to verify not only the system boot but also to check which applications and kernel modules have been loaded in the platform, whether they are expected or undesirable invocations, whether the software has a trusted state, or its configuration is as expected. Consequently, these checks do not necessarily require a new CPU mode of execution or a new operating system but merely rely on the hardware RoT provided by the TPM, which is nowadays ubiquitous in many platforms.

4) REMOTE ATTESTATION TECHNIQUE ADOPTED

Remote attestation [76] is a process through which a remote trusted node (called Verifier) can attest that a device (Prover) is in a legitimate state, i.e., uncompromised. The process exploits an asymmetric challenge/response technique, in which the Verifier generates a challenge (nonce) and sends an attestation request to the *Attester*, or *Prover* (see Fig. 3).

Given that the time unit devices are equipped with a TPM and run a custom Linux distribution, we have considered a

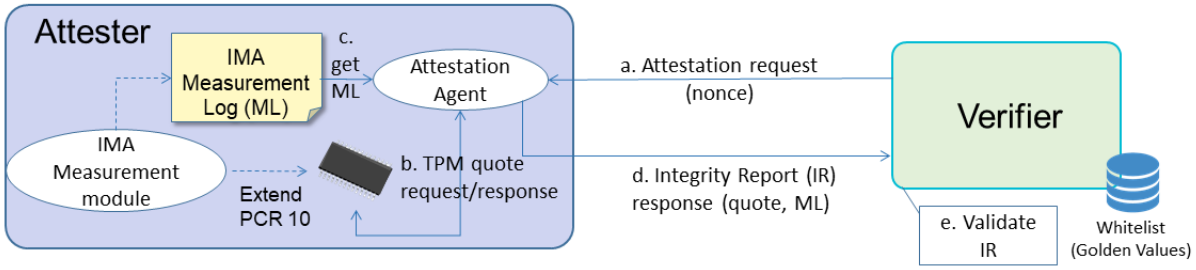

FIGURE 3. Remote attestation.

remote attestation implementation in which the Prover digitally signs the measurements created with IMA by using asymmetric keys generated from the EK inside the TPM residing on the Prover. With TPM, the attestation schemes can ensure the attestation response is trustworthy [77]. To ensure scalability, the entire process is managed by employing the Keylime framework.

Each *Attester* platform, equipped with a TPM 2.0 chip, has its EK key, from which the AK keys may be derived, along with the corresponding certificate EK_{cert} issued by the TPM manufacturer. The AK keys are used for signing the Integrity Reports (IRs). Upon receiving the attestation request, the attestation agent collects the attestation data, signs it with an AK key of the platform by performing a TPM quote operation (signature over the PCR 10 content), then combines it with the nonce and the ML and sends back to the Verifier the IR. The Verifier must be able to verify the AK certificate, which binds the Initial Attestation Key or the Local Attestation Key with the device's identity. Moreover, it must have a so-called *whitelist* holding a set of known-good values compared to the ones in the Attester's ML during the attestation process. Thus, to check the Attester at run time, the Verifier can use PCR 10's content to verify that the ML has not been tampered with. If the ML is valid, the Verifier analyzes it entry by entry by confronting each entry against the ones in the whitelist. This processing allows for determining if the change in the system state represented by each entry in the ML is trustworthy.

To resume, the Verifier performs the following operations: 1) checks the TPM quote signature; 2) validates that the ML entries match the value of PCR 10; 3) checks whether the files in the ML are the same ones as in the Verifier's whitelist (golden values) and their digests must match. A more detailed explanation of this process can be found also in [72] and [78].

B. SCALABLE REMOTE ATTESTATION WITH THE KEYLIME FRAMEWORK

Developed by a security research group in Massachusetts Institute of Technology (MIT) "Lincoln Laboratory" and presented in 2016 through the whitepaper [79], the Keylime framework aims to provide *high scalability* to the remote boot attestation process. Nowadays, Keylime

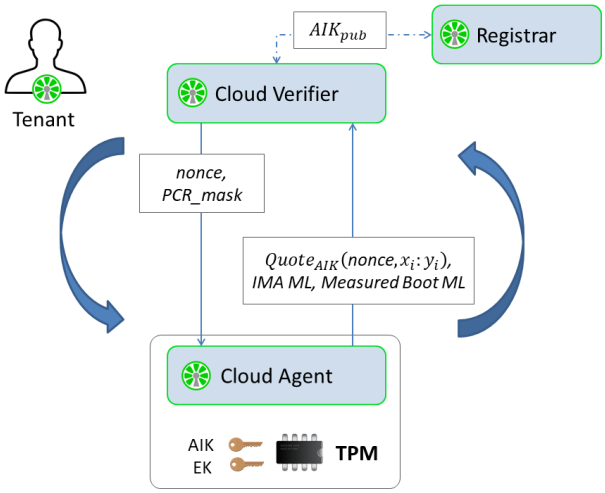

FIGURE 4. Main components of the Keylime architecture employed for scalable remote attestation in the prototyped TDN.

security software is deployed to the IBM cloud, and is a Cloud Native Computing Foundation sandbox technology with more than 30 open-source developers [80]. The Keylime framework was born in the cloud computing context and, more specifically, in the scenarios supporting Infrastructure as a Service (IaaS). In these scenarios, the final user (called Tenant) is provisioned with resources (cloud nodes) that could be physical or virtual machines. The resources are given to the user who will deploy and control his software to these nodes. The key point is that the tenants have no control over the underlying infrastructure. Thus, they cannot ensure, with their own implementation, that the platform given by the IaaS provider remains in a good and safe state during the computation.

The Keylime architecture exploits four main components, as depicted in Fig. 4:

- the *Tenant*: is the module that kicks off the framework; it registers the *Could Agent* with the *Cloud Verifier*, sending it all the information (that is the whitelist, exclude list, and TPM policy) necessary to start the periodic remote attestation on the node. Moreover, it verifies

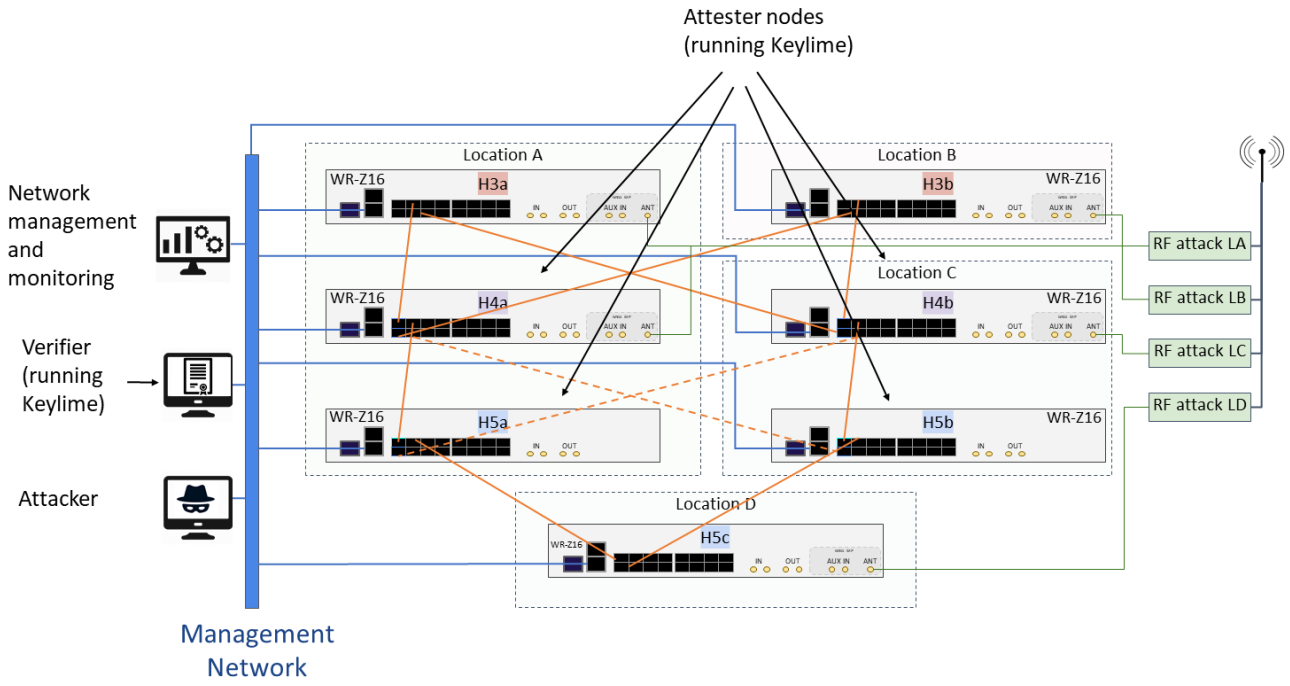

FIGURE 5. Testbed description.

the authenticity of the TPM of the remote platform by checking the validity of the certificate of the TPM's Endorsement Key (EK_{cert}). If the TPM does not have a certificate, the *Cloud Agent* sends the public part of the EK key (EK_{pub}) and the Tenant should verify that the EK_{pub} is among those allowed;

- the *Cloud Agent*: is a service running on the node to be attested (Attester) and is in charge of sending the Integrity Reports (IRs) to the *Cloud Verifier*.
- the *Registrar*: is the component to which every network node has to register via its own UUID, an alphanumerical identifier. For each registered node, the Registrar stores three different pieces of information: the EK_{pub} keys, the AIK_{pub} keys, and the EK_{cert}. In this way, the Tenant can associate an EK_{pub} and an AIK_{pub} to a node, which holds the corresponding EK_{priv} and AIK_{priv}. The verification of the EK_{cert} is fulfilled by the Tenant (not by the Registrar) in the key derivation protocol.
  The Registrar receives the TPM credentials from the Agent and sends them: a) to the Cloud Verifier, to allow it to check the TPM 2.0 quotes signed with the private part of the AIK, namely AIK_{priv}, and b) to the Tenant, to allow it to verify the validity of the EK_{cert}/EK_{pub};
- the *Cloud Verifier*: is the core element of the whole framework. Once a node is registered, the Tenant can start monitoring it by asking the Cloud Verifier to verify the node's integrity state. By default, the Verifier sends every 2 seconds an attestation request to the Cloud agent. However, this value can be changed (because it's configurable) to check the Attester more frequently. To evaluate the integrity level of the node running the Agent, the Verifier processes the IRs based on the information received from the Tenant (e.g., the whitelist, the TPM policy), as well as the data received from the *Registrar* like the AIK_{pub} used for quote' signature verification.

VI. DESCRIPTION OF TESTBED EXPLOITING KEYLIME

We have installed the Keylime components on a testbed TDN exploiting WR-Z16 devices in a controlled environment at Telefonica premises, as shown in Fig. 5. The entire testbed is extensively described in [1]. In the tested scenarios, the attested nodes H4a and H5a acted as primary (time unit) devices, while H4b and H5b acted as backup (secondary) devices. A remote trusted node acted as Verifier in the remote attestation process, while the attacks have been initiated from an Attacker node. We deliberately excluded details about the execution of GNSS attacks (shown as RF attacks in Fig. 5) since they are out of scope in this paper.

To counter the software integrity attacks, we used a software TPM namely IBM's TPM 2.0 simulator [81], which has been installed on the H4a, H4b, H5a, H5b, and H5c devices. More precisely, on the WR-Z16 devices running a customized Linux version with a Xilinx kernel, we installed the following software: IBM's TPM2.0 simulator (ibmtpm1661), Python (version 3.10), TPM-related software like tpm2-tss (version 3.1.0) [82], tpm2-tools (version 4.3.2) [83], and the Keylime

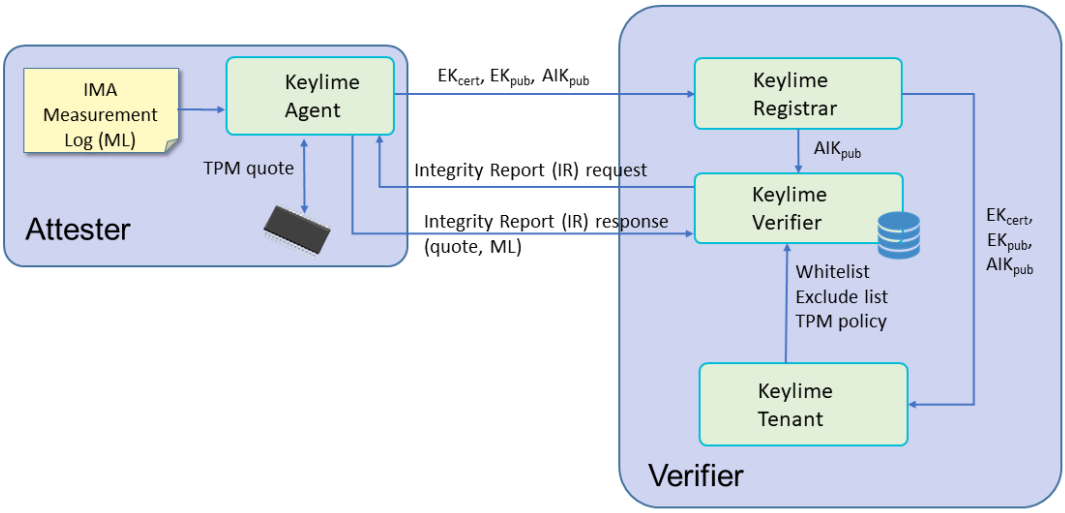

FIGURE 6. Installed Keylime components on the time unit devices (indicated as Attester) and on the trusted remote node (indicated as Verifier).

(Cloud) Agent (version 6.2.1) [84]. On the Verifier machine, running an Ubuntu 20.04 Linux distribution, we installed tpm2-tools (version 5.2), Python (version 3.8.10), and the Registrar, Verifier, and Tenant Keylime components (version 6.2.1), as illustrated in Fig. 6. Additionally, we developed and installed a script developed in Python for generating automatically the whitelist instead of manually editing this file.

VII. EXPERIMENTS

To perform the experiments we have exploited the testbed detailed in Section VI. To modify the daemons or their configuration, we have connected remotely (with OpenSSH[1]) from the Attacker machine to the time devices under attack. We used classical Linux commands (e.g., `mv`, `cp`, or `vi`) to move, copy or edit files.

A. SOFTWARE INTEGRITY ATTACKS: DESCRIPTION

We have considered and tested the following software integrity attacks:

1) SOFTWARE CONFIGURATION MODIFICATION

This test simulates a scenario in which an attacker may gain access to the H4a WR-Z16 unit, e.g., an impostor who manages to steal access credentials for the administrator account or a legitimate administrator behaving maliciously. Subsequently, the attacker changes the configuration file of the WR-PTP daemon, that is, `/root/.config` file. In this way, he can modify a calibration value (such as by adding 100 ns), which results in a change of the 1-PPS offset of the slaves after a reboot. This attack is applied to the H4a node only, but subsequently, it can affect the synchronization of the H5a and H5b nodes.

[1]https://github.com/openssh/openssh-portable

Countermeasure Adopted: The attack is detectable by adding the entry corresponding to the IMA measurement of the abovementioned configuration file in the whitelist of the Verifier. Thus, with the Keylime remote attestation process, any change in the parameters listed in the configuration file would generate a different IMA measurement than the golden value in the whitelist of the Verifier.

2) EXECUTABLE SOFTWARE MODIFICATION

In this case, the attacker wants to modify or replace the WR-PTP software, that is, the `ppsi` daemon. First, we assume the attacker can access with superuser privileges the software installed and running on the H5a WR-Z16 unit. Next, he replaces the executable file of the PPSI daemon (i.e., `/wr/bin/ppsi`) with a malicious file, which will slowly shift the 1-PPS of the H5a node. With a sufficiently long attack duration, the attacker can gradually introduce an arbitrarily large time offset on the 1-PPS output of the target node. We note that the slow shift of the 1-PPS may go "undetected" unless the mitigation solution we propose with Keylime is adopted. Moreover, we remark that `ppsi` is open-source, so any potential attacker with sufficient knowledge can modify it.

Countermeasure Adopted: The attack is detectable by exploiting the Keylime remote attestion process, which periodically checks the IMA measurement of the running WR-PTP software, more precisely, every 2 seconds. As part of the verification steps in the remote attestation process, this measurement is cryptographically checked against the PCR 10 value stored in the WR-Z16's TPM and is confronted with the golden value saved in the whitelist on the Verifier.

3) GNSS RECEIVER CONFIGURATION MODIFICATION

This attack targets the configuration of the GNSS receiver module (i.e., mosaic-T) installed on the H5c WR-Z16 unit.

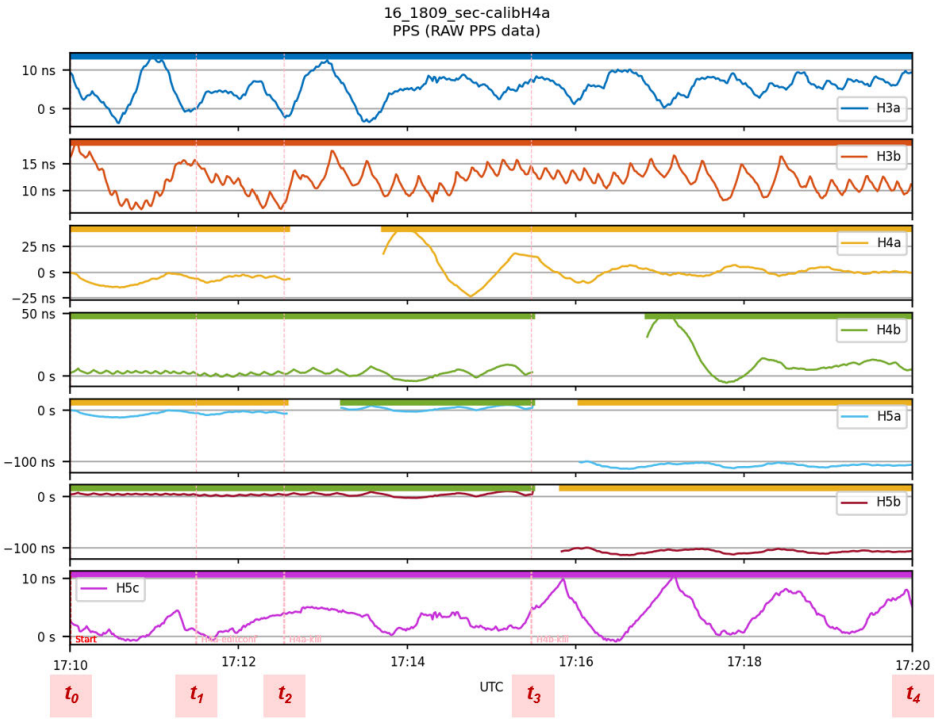

FIGURE 7. Software Configuration Modification (H4-SW-PC) test results.

We assumed that, through privilege escalation, an attacker could gain access to the command interface of the GNSS receiver, i.e., UART. Thus, he could modify the calibration parameters of the receiver. In practice, the attack consists of sending a malicious configuration command (i.e., `setPPSParameters`) to the mosaic-T receiver on the H5c unit, with a wrong calibration parameter for its 1-PPS signal output (i.e., a wrong cable delay). According to the mosaic-T Reference Guide [85], this parameter can be set in a range from -1 ms to $+1$ ms, thus potentially resulting in a considerable time offset on H5c compared to the other nodes on the network.

Countermeasure Adopted: To detect this attack, a dedicated daemon in charge with monitoring and control of the GNSS receiver on the H5c node periodically polls the current GNSS configuration through the specific command `lstConfigFile, Current`. In detail, the daemon retrieves the configuration at bootstrap and stores it in a file, named `nominalGNSSconfig.txt`. Next, it periodically polls the GNSS configuration and stores it in a separate file named `currentGNSSconfig.txt`. The digests of both configuration files have also been added to the Keylime whitelist. In this way, any modification on the GNSS configuration results in a different IMA measurement detectable by Keylime by confronting it with the golden values in the Verifier whitelist.

VIII. EXPERIMENTAL RESULTS

This section describes the experimental results obtained during the test campaign.

A. SOFTWARE CONFIGURATION MODIFICATION

This test (named *H4-SW-PC*) consisted in modifying the configuration file (i.e., `/root/.config`) on the H4a timing device, to change calibration values (by adding 100 ns) that affects the time synchronization of slave nodes (i.e., H5a and H5b nodes).

The test began with all the nodes in nominal conditions (at time t_0). After modifying the H4a configuration file, we restarted both H4a and H4b. Next, we have detected that the 1-PPS output of H4a was shifted by 100 ns.

The results of this test are reported in Fig. 7. In detail, the actions sequentially carried out during the test are given below:

1) At t_1, we modified the calibration value in the `.config` file of H4a. At this point, from the operational point of view, nothing happened, as the daemon has not loaded yet the modified value. Nevertheless, Keylime has already detected the modified calibration value, since the digest of the modified file has a different value than the golden value stored on the Verifier. Thus, an entry indicating the integrity check error is added by the Verifier in a specific Log file.
2) At t_2, we killed the `hald` daemon at the H4a node, forcing the daemon to reload the (modified) calibration value when it has been (automatically) restarted. Since we have killed the daemon in H4a, the H5a loosed its primary timing source (i.e., H4a, shown in yellow in Fig. 7). Thus, it switched to use the H4b node as timing source (whose evolution is shown in green in Fig. 7).

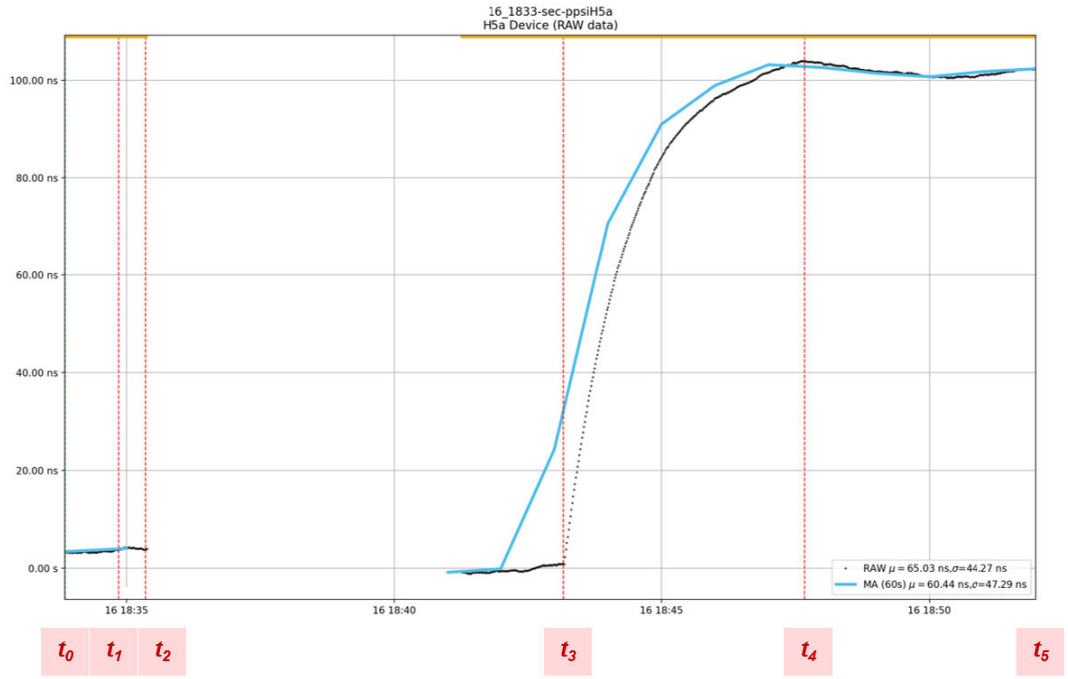

FIGURE 8. Executable Software Modification (H5-SW-PS) test results (raw PPS in black, moving average of 60 s in blue).

3) At t_3, we killed the daemon in the H4b node, so that H5a switched back to using H4a (shown in yellow in Fig. 7) as primary timing source. At this point, the H5b node also loosed its primary reference (H4b) so it switched to the H4a node. Finally, when both the H5a and H5b nodes reconnected to the network, they had a 100 ns offset with respect to the unaffected nodes, namely the H3a, H3b, and H5c nodes, until the end of the test at t_4.

Each of the above actions is represented by a pink dashed vertical line in Fig. 7, by using the UTC timescale.

We highlight that the Keylime Verifier has signaled already at the time t_1 that the configuration file has been modified. Thus, it allows early detection of this attack before any measurable impact on the synchronization of the nodes could occur.

We also mention that in the WR-Z16 units, the conventional Best Master Clock (BMC) algorithm has been replaced by a custom Failover algorithm (i.e., FOCA) to enhance security and robustness. The FOCA algorithm [1] provides an extra layer of protection against this type of attack, as when a failure is detected on H4a (i.e., killing H4a `hald` daemon to reload modified calibration value), the node following H4a will switch to H4b and will not return to H4a unless there is also a failure on H4b. In future firmware releases of WR-Z16, the device will be able to detect that the offset from H4a has been shifted by 100 ns after being available again (e.g., at 17:13:41 in Fig. 7) and will discard this source from being selected.

B. EXECUTABLE SOFTWARE MODIFICATION

This test (named *H5-SW-PS*) is somehow similar to the previous one (i.e., *H4-SW-PC*), but it emulates a malicious modification of the WR-PTP software (that is, the `ppsi` daemon). We had killed the authentic `ppsi` daemon to force the node to reload its modified version.

Fig. 8 shows the results of this test for the H5a node, since this attack only affects the H5a device and no other devices.[2] This graph shows the RAW PPS data (in black) and its corresponding moving average of 60 seconds (in blue).

This test has also started with all the nodes in nominal conditions (at time t_0). The following actions (represented by pink dashed vertical lines in Fig. 8) have been carried out during the test execution:

1) At t_1, we replaced the authentic `ppsi` daemon with a modified version obtained by changing the original `ppsi` source code to introduce some new instructions to read a value in picoseconds from a file and append it at each iteration to the offset. In this way, this altered version may introduce an arbitrary synchronization error on the node.
2) At t_2, we rebooted the H5a time unit device to simulate an "external maintenance". This action was done shortly (i.e., a few seconds) after uploading a modified version of `ppsi`. Nevertheless, it is realistic to imagine an attacker waiting for days or (even) months until an operator reboots this device. After rebooting, the H5a device reconnected itself to H4a. However, as expected, it used the modified `ppsi` daemon instead of the authentic one. At this point, although the attacker has completed the first attack step, no measurable

[2]This was done for simplicity/clarity, but one could also modify the `ppsi` running at master that could affect synchronization of all devices below it but not himself.

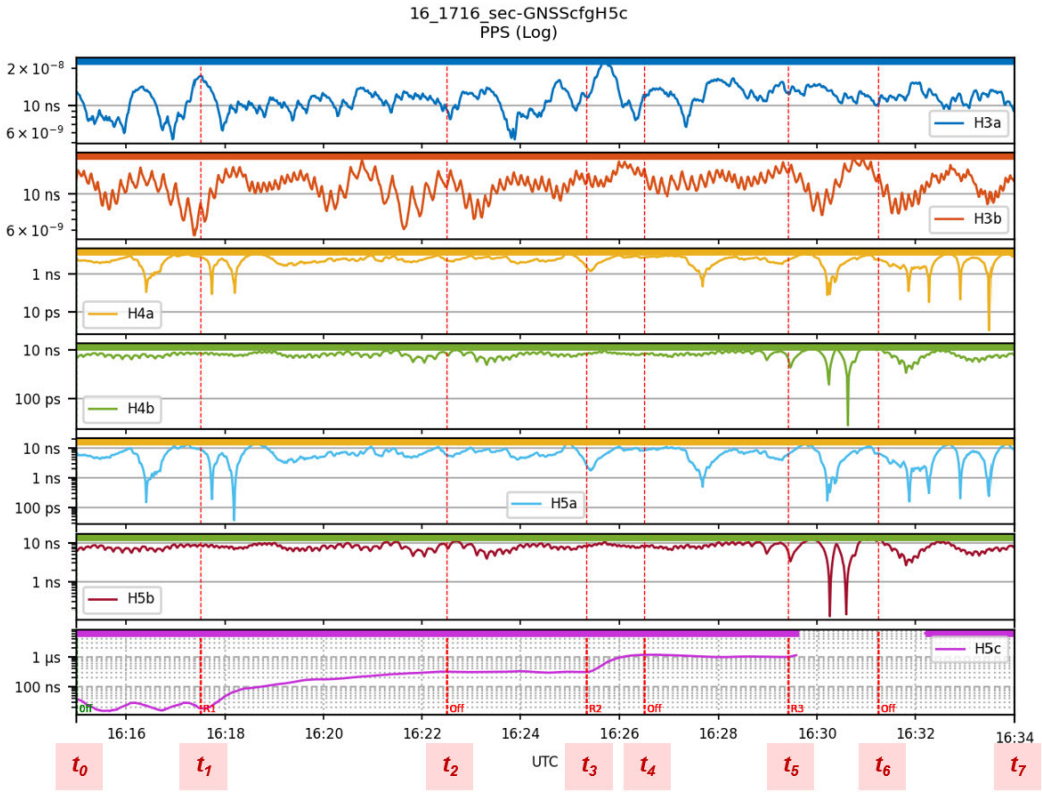

FIGURE 9. GNSS Receiver Configuration Modification (H5-SW-GC) test results ($\log_{10}$ scale on y-axis).

impact has been observed on the synchronization of the device. The modified `ppsi` daemon was waiting for new inputs from the attacker.

3) At t_3, we introduced a modest drift at each iteration (once per second).
4) At t_4, we stopped the drift, reaching a stable offset of 100 ns on the H5a node until the end of the test at t_5.

During this test, the Verifier node detected at t_1 the changes on (the digest of) the `ppsi` daemon at H5a. Thus, it discarded this device from its trusted list. However, without Keylime verification, an attacker could successfully introduce an arbitrary synchronization offset, without any alert in the system. So, this particular attack was **fully** countered with the proposed Keylime-based solution. Without monitoring and detection with Keylime, this attack would remain undetected. In this case, the protection of the software deamon with TC and Keylime technique was the only effective method to counter this insidious attack.

C. GNSS RECEIVER CONFIGURATION MODIFICATION

This test (H5-SW-GC) focused on the mosaic-T GNSS receiver embedded in the H5c node. The attack consisted in slightly changing receiver's configuration parameter related to the calibration delay of the GNSS antenna so that the 1-PPS output has been smoothly shifted from its reference time source. To make the attack scenario more realistic, we have defined three different rates (i.e., $R_1 = 1$ ns/s, $R_2 = 15$ ns/s, $R_3 = 100$ ns/s) to achieve a smooth drift from the reference clock.

Fig. 9 reports the results of this test, using a $\log_{10}$ scale on the y-axis for H4a, H4b, H5a, H5b, and H5c nodes. In this way, it is possible to appreciate the different rates and the total synchronization offset reached by the H5c node at the end of the test.

For this attack, the test has started also with all the nodes in nominal conditions (at time t_0), and the following actions, represented by pink dashed vertical lines in Fig. 9, have been carried out during the test execution:

1) At t_1, we started the attack (simulation) on H5c by increasing the antenna delay with the initial rate R_1 (1 ns/s). This modest rate resulted in the introduction of a slowly increasing offset, which was difficult to detect.
2) At t_2, we stopped the offset increase of the antenna delay.
3) At t_3, we resumed the attack on H5c antenna delay, but using a larger increase rate (i.e., $R_2 = 15$ ns/s). This increased rate was close to the boundary for which the WR-Z16 oscillator could track the changes, while higher rates could result in instability or a loss of lock.
4) At t_4, we stopped again the offset increasing.
5) At t_5, the attack was performed with a very high (unrealistic) rate of increase (i.e., $R_3 = 100$ ns/s).

It is possible to appreciate in Fig. 9 that the internal oscillator of the H5c cannot follow this high increase rate because it would result in a loss of lock, making this part of the attack immediately detectable by the device.

6) At t_6, we stopped again the offset increasing, and the H5c device synchronized once more its oscillator to the GNSS receiver output, resulting in an offset of more than 1 μs until the end of the test at t_7 (i.e., out of scale in Fig. 9).

During this test, the Verifier detected the changes on the configuration of the mosaic-T and signaled the event via messages (e.g., 'Hashes for file /root/currentGNSSconfig.txt dont'match f98281164d55…') in a Log file generated on the Verifier node.

At t_1 (i.e., 16:17:31.336), the digest of the configuration file (`currentGNSSconfig.txt`) did not match the golden value in the whitelist. Thus, the Verifier immediately stopped validating the H5c device. In this way, the ongoing attack was detected at t_1, corresponding to the first attack stage, before any measurable impact on the synchronization occurred.

It is also worth mentioning that, in the actual configuration of the Verifier, when a device is not valid anymore, the Verifier stops polling it until the Keylime service is restarted. For this reason, we did not reboot the devices during this attack simulation, as we were aware that no further messages would have been logged (unless Keylime is restarted) by changing the rate to another value.

IX. CONCLUSION AND FUTURE WORK

With the rise of 5G networks, mobile telecom operators look for finer time accuracy for synchronization between RAN nodes. At the same time, more and more applications ask for stringent time synchronization in the range of a few microseconds or even tens of nanoseconds. TDNs may respond to such time requirements by employing specialized time device units and specific timing protocols. Nonetheless, TDNs must also be protected against cybersecurity attacks since their components and the provided service represent an appealing target for internal and external attackers.

We considered a TDN exploiting custom-tailored WR-Z16 time unit devices (acting as C-GMCs and D-GMCs) and the WR-PTP protocol for distributing ns-level timing information. We reviewed the attack types applying to TDNs. Then, we focused on the software integrity attacks in which an adversary could compromise the software running on the time unit devices, or its configuration. We designed a TC-enabled solution exploiting the TPM on the time unit devices, a remote trusted node for monitoring the time devices, and the remote attestation protocol supported by the Keylime framework. With this solution, we successfully detected three types of software integrity attacks involving the (software) daemons or configuration of time devices. Through experiments, we showed the effectiveness of this solution in mitigating all the tested attacks, some of which would have remained undetected otherwise.

Future work could address considerable more intricate combined attacks applicable to the network management, time distribution, hardware and software TDN views, and the relative costs needed to activate them.

ACKNOWLEDGMENT

The authors would like to thank all the partners of the ROOT project for the fruitful collaboration leading to the achievements of the results described in this article. Additionally, they would like to thank the anonymous reviewers of IEEE *Access* journal, for providing useful comments to improve this article.

REFERENCES

[1] A. Minetto, B. Rat, M. Pini, B. Polidori, I. De Francesca, L. M. Contreras, and F. Dovis, "Nanosecond-level resilient GNSS-based time synchronization in telecommunication networks through WR-PTP HA," *TechRxiv*, Feb. 2023, doi: 10.36227/techrxiv.22032446.v1.

[2] S. Ruffini, M. Johansson, B. Pohlman, and M. Sandgren. (2021). 5G syncronization requirements and solutions. Ericsson Technology Review. Accessed: Dec. 12, 2022. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-synchronization-requirements-and-solutions

[3] Council of the European Union. (Dec. 2008). *Council Directive 2008/114/EC of 8 December 2008 on the Identification and Designation of European Critical Infrastructures and the Assessment of the Need to Improve Their Protection*. Accessed: Dec. 12, 2022. [Online]. Available: https://eur-lex.europa.eu/eli/dir/2008/114/oj

[4] E. Falletti, D. Margaria, G. Marucco, B. Motella, M. Nicola, and M. Pini, "Synchronization of critical infrastructures dependent upon GNSS: Current vulnerabilities and protection provided by new signals," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2118–2129, Sep. 2019, doi: 10.1109/JSYST.2018.2883752.

[5] *Information Technology—Trusted Platform Module—Part 1: Overview*, Standard ISO/IEC 11889-1:2015. Accessed: Apr. 18, 2023. [Online]. Available: https://www.iso.org/standard/50970.html

[6] F. Girela-López, J. López-Jiménez, M. Jiménez-López, R. Rodríguez, E. Ros, and J. Díaz, "IEEE 1588 high accuracy default profile: Applications and challenges," *IEEE Access*, vol. 8, pp. 45211–45220, 2020, doi: 10.1109/ACCESS.2020.2978337.

[7] M. Lipinski, T. Wlostowski, J. Serrano, and P. Alvarez, "White rabbit: A PTP application for robust sub-nanosecond synchronization," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization for Meas., Control Commun.*, Sep. 2011, pp. 25–30, doi: 10.1109/ISPCS.2011.6070148.

[8] D. G. Berbecaru and S. Sisinni, "Counteracting software integrity attacks on IoT devices with remote attestation: A prototype," in *Proc. 26th Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2022, pp. 380–385, doi: 10.1109/ICSTCC55426.2022.9931765.

[9] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart., 2016.

[10] K. Boyle. *5G is All in the Timing*. Accessed: Dec. 12, 2022. [Online]. Available: https://www.ericsson.com/en/blog/2019/8/what-you-need-to-know-about-timing-and-sync-in-5G-transport-networks

[11] M. Pini, A. Minetto, A. Vesco, D. Berbecaru, L. M. C. Murillo, P. Nemry, I. De Francesca, B. Rat, and K. Callewaert, "Satellite-derived time for enhanced telecom networks synchronization: The ROOT project," in *Proc. IEEE 8th Int. Workshop Metrol. Aerosp. (MetroAeroSpace)*, Jun. 2021, pp. 288–293, doi: 10.1109/MetroAeroSpace51421.2021.9511780.

[12] D. Mills, U. Delaware, J. Martin, J. Burbank, and W. Kasch, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, document IETF RFC 5905, Jun. 2010.

[13] *Time and Phase Synchronization Aspects of Telecommunication Networks*, document ITU-TR G.8271/Y.1366 03/20. Accessed: Dec. 12, 2022. [Online]. Available: https://www.itu.int/rec/T-REC-G.8271-202003-I/en

[14] *Timing Characteristics of Primary Reference Time Clocks*, document ITU-TR G.8272/Y.1367 11/18. Accessed: Feb. 26, 2023. [Online]. Available: https://www.itu.int/rec/T-REC-G.8272-201811-I/en

[15] *Timing Characteristics of Enhanced Primary Reference Time Clock*, document ITU-T R G.8272.1/Y.1367.1 11/16. Accessed: Feb. 26, 2023. [Online]. Available: https://www.itu.int/rec/T-REC-G.8272.1/en

[16] *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2008 (Revision of IEEE Standard 1588-2002), 2008, pp. 1–300, doi: 10.1109/IEEESTD.2008.4579760.

[17] *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2019 (Revision IEEE Standard 1588-2008), 2020, pp. 1–499, doi: 10.1109/IEEESTD.2020.9120376.

[18] P. V. Estrela, S. Neusüß, and W. Owczarek, "Using a multi-source NTP watchdog to increase the robustness of PTPv2 in financial industry networks," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization for Meas., Control, Commun. (ISPCS)*, Sep. 2014, pp. 87–92, doi: 10.1109/ISPCS.2014.6948697.

[19] F. M. Anwar and M. Srivastava, "Applications and challenges in securing time," in *Proc. UNIX CSET Workshop*, 2019, pp. 1–5. [Online]. Available: https://www.usenix.org/conference/cset19/presentation/anwar

[20] D. Berbecaru, A. Lioy, and C. Cameroni, "Supporting authorize-then-authenticate for Wi-Fi access based on an electronic identity infrastructure," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 11, no. 2, pp. 34–54, 2020, doi: 10.22667/JOWUA.2020.06.30.034.

[21] D. Berbecaru, A. Atzeni, M. De Benedictis, and P. Smiraglia, "Towards stronger data security in an eID management infrastructure," in *Proc. 25th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Saint Petersburg, Russia, 2017, pp. 391–395, doi: 10.1109/PDP.2017.90.

[22] T. Mizrahi, *Security Requirements of Time Protocols in Packet Switched Networks*, document IETF RFC 7384, Oct. 2014.

[23] D. Berbecaru, "On creating digital evidence in IP networks with nettrack," in *Handbook of Research on Network Forensics and Analysis Techniques*. Hershey, PA, USA: IGI Global, 2018, doi: 10.4018/978-1-5225-4100-4.ch012.

[24] K. Samalla and P. N. Kumar, "A novel study and analysis on global navigation satellite system threats and attacks," in *High Performance Computing and Networking* (Lecture Notes in Electrical Engineering), vol. 853, C. Satyanarayana, D. Samanta, X. Z. Gao, R. K. Kapoor, Eds. Singapore: Springer, 2022, pp. 371–381, doi: 10.1007/978-981-16-9885-9_31.

[25] D. Margaria, B. Motella, M. Anghileri, J. Floch, I. Fernandez-Hernandez, and M. Paonni, "Signal structure-based authentication for civil GNSSs: Recent solutions and perspectives," *IEEE Signal Process. Mag.*, vol. 34, no. 5, pp. 27–37, Sep. 2017, doi: 10.1109/MSP.2017.2715898.

[26] I. Fernández-Hernández, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodríguez, and J. D. Calle, "A navigation message authentication proposal for the Galileo open service," *Navigation*, vol. 63, no. 1, pp. 85–102, Mar. 2016, doi: 10.1002/navi.125.

[27] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert, "Impact of cyberattacks on precision time protocol," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 5, pp. 2172–2181, May 2020, doi: 10.1109/TIM.2019.2918597.

[28] M. Dalmas, H. Rachadel, G. Silvano, and C. Dutra, "Improving PTP robustness to the Byzantine failure," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Beijing, China, Oct. 2015, pp. 111–114, doi: 10.1109/ISPCS.2015.7324693.

[29] E. Itkin and A. Wool, "A security analysis and revised security extension for the precision time protocol," *IEEE Trans. Depend. Sec. Comput.*, vol. 17, no. 1, pp. 22–34, Jan. 2020, doi: 10.1109/TDSC.2017.2748583.

[30] W. Alghamd and M. Schukat, "A detection model against precision time protocol attacks," in *Proc. 3rd Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)*, Riyadh, Saudi Arabia, Mar. 2020, pp. 1–3, doi: 10.1109/ICCAIS48893.2020.9096742.

[31] S. Barreto, A. Suresh, and J.-Y. Le Boudec, "Cyber-attack on packet-based time synchronization protocols: The undetectable delay box," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, May 2016, pp. 1–6, doi: 10.1109/I2MTC.2016.7520408.

[32] *The ROOT (Rolling Out OSNMA for the Secure Synchronisation of Telecom Networks) Project*. Accessed: Dec. 12, 2022. [Online]. Available: https://www.gnss-root.eu/

[33] European Global Navigation Satellite System Agency. (Feb. 2021). *Tests of Galileo OSNMA Underway*. Accessed: Dec. 12, 2022. [Online]. Available: https://www.gsa.europa.eu/newsroom/news/tests-galileo-osnma-underway

[34] Seven Solutions. (2022). *WR-Z16 the Reliable Precise Time Fan-Out for White Rabbit Distribution on 1G Ethernet-Based Networks*. Accessed: Apr. 18, 2023. [Online]. Available: https://sevensols.com/wr-z16/

[35] *TCG Software Stack (TSS) Specification*. Accessed: Feb. 25, 2023. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-software-stack-tss-specification/

[36] *Integrity Measurement Architecture (IMA)*. Accessed: Feb. 26, 2023. [Online]. Available: https://sourceforge.net/projects/linux-ima/

[37] *Keylime. Bootstrap & Maintain Trust on the Edge/Cloud and IoT*. Accessed: Dec. 12, 2022. [Online]. Available: https://keylime.dev/

[38] W. Alghamdi and M. Schukat, "Precision time protocol attack strategies and their resistance to existing security extensions," *Cybersecurity*, vol. 4, no. 1, pp. 1–17, Dec. 2021, doi: 10.1186/s42400-021-00080-y.

[39] W. Alghamdi and M. Schukat, "Practical implementation of APTs on PTP time synchronisation networks," in *Proc. 31st Irish Signals Syst. Conf. (ISSC)*, Jun. 2020, pp. 1–5, doi: 10.1109/ISSC49989.2020.9180157.

[40] W. Alghamdi and M. Schukat, "Cyber attacks on precision time protocol networks—A case study," *Electronics*, vol. 9, no. 9, p. 1398, Aug. 2020, doi: 10.3390/electronics9091398.

[41] W. Alghamdi and M. Schukat, "A security enhancement of the precision time protocol using a trusted supervisor node," *Sensors*, vol. 22, no. 10, p. 3671, May 2022, doi: 10.3390/s22103671.

[42] A. Beuhring and K. Salous, "Beyond blacklisting: Cyberdefense in the era of advanced persistent threats," *IEEE Secur. Privacy*, vol. 12, no. 5, pp. 90–93, Sep. 2014, doi: 10.1109/MSP.2014.86.

[43] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019, doi: 10.1109/COMST.2019.2891891.

[44] W. Alghamdi and M. Schukat, "Advanced methodologies to deter internal attacks in PTP time synchronization networks," in *Proc. 28th Irish Signals Syst. Conf. (ISSC)*, Killarney, Ireland, Jun. 2017, pp. 1–6, doi: 10.1109/ISSC.2017.7983636.

[45] D. G. Berbecaru and A. Lioy, "Attack strategies and countermeasures in transport-based time synchronization solutions," in *Intelligent Distributed Computing XIV* (Studies in Computational Intelligence), vol. 1026, D. Camacho, D. Rosaci, G. M. L. Sarné, and M. Versaci, Eds. Cham, Switzerland: Springer, 2022, doi: 10.1007/978-3-030-96627-0_19.

[46] D. Margaria and A. Vesco, "Trusted GNSS-based time synchronization for industry 4.0 applications," *Appl. Sci.*, vol. 11, no. 18, p. 8288, Sep. 2021, doi: 10.3390/app11188288.

[47] T. Mizhari, *Security Requirements of Time Protocols in Packet Switched Networks*, document RFC 7384, Oct. 2014. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7384

[48] P. Kemparaj and S. S. Kumar, "Secure precision time protocol in packet switched networks," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control, Commun. (ISPCS)*, Portland, OR, USA, Sep. 2019, pp. 1–6, doi: 10.1109/ISPCS.2019.8886643.

[49] M. Langer, K. Teichel, D. Sibold, and R. Bermbach, "Time synchronization performance using the network time security protocol," in *Proc. Eur. Freq. Time Forum (EFTF)*, Turin, Italy, Apr. 2018, pp. 138–144, doi: 10.1109/EFTF.2018.8409017.

[50] R. Annessi, J. Fabini, and T. Zseby, "It's about time: Securing broadcast time synchronization with data origin authentication," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Vancouver, BC, Canada, Jul. 2017, pp. 1–11, doi: 10.1109/ICCCN.2017.8038418.

[51] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: Software-based attestation for embedded devices," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2004, pp. 272–282, doi: 10.1109/SECPRI.2004.1301329.

[52] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann, "PUFatt: Embedded platform attestation based on novel processor-based PUFs," in *Proc. 51st Annu. Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2014, pp. 1–6, doi: 10.1145/2593069.2593192.

[53] H. Tan, W. Hu, and S. Jha, "A remote attestation protocol with trusted platform modules (TPMs) in wireless sensor networks," *Secur. Commun. Netw.*, vol. 8, no. 13, pp. 2171–2188, Sep. 2015, doi: 10.1002/sec.1162.

[54] *Intel Trusted Execution Technology Enabling Guide*. Accessed: May 19, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/txt-enabling-guide.html
[55] I. Sfyrakis and T. Gross, "A survey on hardware approaches for remote attestation in network infrastructures," 2020, *arXiv:2005.12453*.
[56] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, and S. Ranise, "Collective remote attestation at the Internet of Things scale: State-of-the-art and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2447–2461, 4th Quart., 2020, doi: 10.1109/COMST.2020.3008879.
[57] C. Günther, "A survey of spoofing and counter-measures," *Navigat., J. Inst. Navigat.*, vol. 61, no. 3, pp. 159–177, Sep. 2014, doi: 10.1002/navi.65.
[58] D. Margaria, G. Marucco, and M. Nicola, "A first-of-a-kind spoofing detection demonstrator exploiting future Galileo E1 OS authentication," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Apr. 2016, pp. 442–450, doi: 10.1109/PLANS.2016.7479732.
[59] D. Borio, F. Dovis, H. Kuusniemi, and L. Lo Presti, "Impact and detection of GNSS jammers on consumer grade satellite navigation receivers," *Proc. IEEE*, vol. 104, no. 6, pp. 1233–1245, Jun. 2016, doi: 10.1109/JPROC.2016.2543266.
[60] M. Ullmann and M. Vögeler, "Delay attacks—Implication on NTP and PTP time synchronization," in *Proc. Int. Symp. Precis. Clock Synchronization Meas., Control Commun.*, Oct. 2009, pp. 1–6, doi: 10.1109/ISPCS.2009.5340224.
[61] T. Ylonen and C. Lonvick, *The Secure Shell (SSH) Connection Protocol*, document IETF RFC 4254, 2006.
[62] E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.3*, document IETF RFC 8446, 2018.
[63] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2027–2051, 3rd Quart., 2016, doi: 10.1109/COMST.2016.2548426.
[64] D. Berbecaru and A. Lioy, "On the robustness of applications based on the SSL and TLS security protocols," in *Public Key Infrastructure* (Lecture Notes in Computer Science), vol. 4582, J. Lopez, P. Samarati, and J. L. Ferrer, Eds. Berlin, Germany: Springer, 2007, doi: 10.1007/978-3-540-73408-6_18.
[65] D. G. Berbecaru and G. Petraglia, "TLS-monitor: A monitor for TLS attacks," in *Proc. IEEE 20th Consum. Commun. Netw. Conf. (CCNC), 5th Int. Workshop Secur. Trust Privacy Cyber-Phys. Syst. (STP-CPS)*, Las Vegas, NV, USA, Jan. 2023, pp. 8–11.
[66] Y. Guan and X. Ge, "Distributed attack detection and secure estimation of networked cyber-physical systems against false data injection attacks and jamming attacks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 1, pp. 48–59, Mar. 2018, doi: 10.1109/TSIPN.2017.2749959.
[67] (Jun. 2015). *Precision Time Protocol Daemon (1588–2008)*. Accessed: Dec. 12, 2022. [Online]. Available: https://manpages.debian.org/stretch/ptpd/ptpd.8.en.html
[68] (Mar. 2021). *The GPSD Project. GPSD(8) Manual Page*. Accessed: Dec. 12, 2022. [Online]. Available: https://gpsd.io/gpsd.html
[69] (Aug. 2014). *PPSi: PTP Ported to Silicon, Wiki—Open Hardware Repository*. Accessed: Dec. 12, 2022. [Online]. Available: https://ohwr.org/project/ppsi/wikis/home
[70] S. F. J. J. Ankergård, E. Dushku, and N. Dragoni, "State-of-the-art software-based remote attestation: Opportunities and open issues for Internet of Things," *Sensors*, vol. 21, no. 5, p. 1598, Feb. 2021, doi: 10.3390/s21051598.
[71] W. Arthur and D. Challener, *A Practical Guide to TPM 2.0*. New York, NY, USA: Apress, 2015.
[72] S. Sisinni, "Verification of software integrity in distributed systems," M.S. thesis, Politecnico di Torino, Torino, Italy, 2021. [Online]. Available: https://webthesis.biblio.polito.it/20403/1/tesi.pdf
[73] *Trusted Computing Group Trusted Platform Module Library Part 1: Architecture*, TCG Published, New Delhi, India, Nov. 2019.
[74] *TPM 2.0 Library*. Accessed: May 19, 2023. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/
[75] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proc. 13th USENIX Secur. Symp. (USENIX Security)*, San Diego, CA, USA, Aug. 2004, pp. 223–238. Accessed: May 19, 2023. [Online]. Available: https://www.usenix.org/legacy/publications/library/proceedings/sec04/tech/full_papers/sailer/sailer.pdf
[76] B. Kuang, A. Fu, W. Susilo, S. Yu, and Y. Gao, "A survey of remote attestation in Internet of Things: Attacks, countermeasures, and prospects," *Comput. Secur.*, vol. 112, Jan. 2022, Art. no. 102498, doi: 10.1016/j.cose.2021.102498.
[77] W. Xu, X. Zhang, H. Hu, G.-J. Ahn, and J.-P. Seifert, "Remote attestation with domain-based integrity model and policy analysis," *IEEE Trans. Depend. Sec. Comput.*, vol. 9, no. 3, pp. 429–442, May 2012, doi: 10.1109/TDSC.2011.61.
[78] S. Sisinni, D. Margaria, I. Pedone, A. Lioy, and A. Vesco, "Integrity verification of distributed nodes in critical infrastructures," *Sensors*, vol. 22, no. 18, p. 6950, Sep. 2022, doi: 10.3390/s22186950.
[79] N. Schear, P. T. Cable, T. M. Moyer, B. Richard, and R. Rudd, "Bootstrapping and maintaining trust in the cloud," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, New York, NY, USA, Dec. 2016, pp. 65–77, doi: 10.1145/2991079.2991104.
[80] K. Foy. (Jul. 27, 2021). *Keylime Security Software is Deployed to IBM Cloud*. Accessed: May 9, 2023. [Online]. Available: https://news.mit.edu/2021/keylime-security-software-deployed-ibm-cloud-0727
[81] *IBM's Software TPM 2.0*. Accessed: May 9, 2023. [Online]. Available: https://sourceforge.net/projects/ibmswtpm2/
[82] *Linux TPM2 & TSS2 Software, TPM2-TSS (Version 3.1.0)*. Accessed: May 9, 2023. [Online]. Available: https://github.com/tpm2-software/tpm2-tss.git
[83] *Linux TPM2 & TSS2 Software, TPM2-Tools*. Accessed: May 9, 2023. [Online]. Available: https://github.com/tpm2-software/tpm2-tools.git
[84] *Keylime V6.2.1*. Accessed: May 9, 2023. [Online]. Available: https://github.com/keylime/keylime.git
[85] *Mosaic-T Reference Guide, Version 4.10.0*, Septentrio, Leuven, Belgium, Jun. 2021.
[86] U. Kröner, C. Bergonzi, J. Fortuny-Guasch, R. Giuliani, F. Littmann, D. Shaw, and D. Symeonidis. (2010). *Hardening of GNSS Based Trackers*. Accessed: Dec. 12, 2022. [Online]. Available: https://publications.jrc.ec.europa.eu/repository/bitstream/JRC58733/reqno_jrc58733_st_report_on_hardening_of_gnss_based_trackers_release_final.pdf

DIANA GRATIELA BERBECARU (Member, IEEE) received the M.Sc. degree in computer science and engineering from the University of Craiova, Romania, and the Ph.D. degree in computer science from Politecnico di Torino, Italy. She is currently an Assistant Professor with the Department of Control and Computer Engineering (DAUIN), Politecnico di Torino. Her research interests include X.509 certificates, digital identities, privacy, authentication, TLS protocol, cyber-attacks, time synchronization, and trusted computing. She is a member of the TORSEC Cybersecurity Research Group.

SILVIA SISINNI received the M.Sc. degree in computer engineering from Politecnico di Torino. She is currently pursuing the Ph.D. degree in computer engineering. Her current research interests include trusted execution environments, trusted computing, trusted channels, and confidential computing. She is a member of the TORSEC Cybersecurity Research Group.

ANTONIO LIOY received the M.Sc. degree (summa cum laude) in electronic engineering and the Ph.D. degree in computer engineering from Politecnico di Torino. He is currently a Full Professor with Politecnico di Torino, where he leads the TORSEC Cybersecurity Research Group. His research interests include network security, policy-based system protection, trusted computing, and electronic identity.

BENOIT RAT received the M.Sc. degree in communication systems from the Swiss Federal Institute of Technology in Lausanne (EPFL), in 2008. He joined Seven Solutions (acquired by Orolia, in 2021) as an Embedded Software Developer. In 2010, he started collaborating with the CERNs Timing Group on the development of the White Rabbit Technology and since, he has been continued bringing sub-nanosecond synchronization (PTP-HA v2019) to a wide range of devices in time-critical infrastructures. He is currently a Solution Architect and also responsible of identifying market needs and trends (i.e. fintech, datacenters, and telecom) and to design and deploy innovative solutions.

DAVIDE MARGARIA received the B.Sc. and M.Sc. degrees in telecommunication engineering from Politecnico di Torino, in 2003 and 2007, respectively. He is currently a Senior Researcher with the Connected Systems and Cybersecurity research domain of the LINKS Foundation. Throughout his career, he has also held a teaching role with Politecnico di Torino, as a Lecturer under grant. He is coauthor of more than 60 papers on peer-reviewed scientific journals and conference proceedings. His current research interests include cybersecurity solutions for cyber-physical systems in the industry 4.0 paradigm and on cryptographic protocols for zero-knowledge proof (ZKP) and verifiable credentials (VCs). His past research interests included Galileo and modernized GPS receivers, especially innovative signal processing strategies, GNSS authentication techniques, and the mitigation of multipath, jamming, and spoofing signals.

ANDREA VESCO received the M.Sc. degree in telecommunication engineering and the Ph.D. degree in computer and system engineering from Politecnico di Torino, in 2003 and 2009, respectively. After one year of Postdoctoral Researcher with the Control and Computer Engineering Department, as a member of the Computer Networks Group, Politecnico di Torino, he joined the Networking Laboratory, Istituto Superiore Mario Boella (ISMB), in 2010. He is currently with the LINKS Foundation a non-profit institution promoting, conducting and strengthening innovation, research processes and the technology transfer. He leads the cybersecurity research team focusing on the security of connected systems, such as the IoT and critical infrastructures. His research interests include the Quality of Service (QoS) over packet switched networks and wireless access networks. Moreover he carried out research activities on the QoS over Network on Chip (NoC) in collaboration with the System-Level Design Group, Columbia University of the city of New York. He had the opportunity to focus his researches on digital smart cities, from 2013 to 2018.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement