Trusted GNSS-Based Time Synchronization for Industry 4.0 Applications

MDPI

Article

Trusted GNSS-based Time Synchronization for Industry 4.0 Applications

Davide Margaria [1,*,‡] and Andrea Vesco [1,‡]

[1] LINKS Foundation, Torino, Italy; firstname.lastname@linksfoundation.com
* Correspondence: davide.margaria@linksfoundation.com
‡ These authors contributed equally to this work.

Abstract: The protection of satellite-derived timing information is becoming a fundamental requirement in Industry 4.0 applications, as well as in a growing number of critical infrastructures. All the industrial systems where several nodes or devices communicate and/or coordinate their functionalities by means of a communication network need accurate, reliable and trusted time synchronization. For instance, the correct operation of automation and control systems, measurement and automatic test systems, power generation, transmission, and distribution typically require a sub-microsecond time accuracy. This paper analyses the main attack vectors and stresses the need for software integrity control at network nodes of Industry 4.0 applications to complement existing security solutions that focus on GNSS RF Spectrum and Precise Time Protocol (PTP), also known as IEEE-1588. A real implementation of a Software Integrity Architecture in accordance with Trusted Computing principles concludes the work together with the presentation of promising results obtained with a flexible and reconfigurable testbed for hands-on activities.

Keywords: trusted computing; industry 4.0; cyber-physical system; time synchronization; cybersecurity; global navigation satellite system; embedded system

Citation: Margaria, D.; Vesco, A. Trusted GNSS-based Time Synchronization for Industry 4.0 Applications. *Appl. Sci.* **2021**, *1*, 0. https://doi.org/

Academic Editor: Silvio Abrate

Received:
Accepted:
Published:

1. Introduction

The cornerstone of Industry 4.0 [1] are Cyber-Physical Systems (CPS) which are closely connected with computer systems and which can interact and collaborate with other CPS systems [2]. These concepts represent the basis of decentralization and cooperation between connected systems: they are closely linked to the Industry 4.0 paradigm and fully enabled by Information and Communication Technologies [3–5].

Networked environments and Industrial Internet of Things (IIoT) [4] need accurate, reliable and trusted time synchronization and time distribution [6], with different requirements in terms of synchronization accuracy and precision. For instance, the scheduling and synchronization of multiple factories represents an interesting high level application of the Industry 4.0 paradigm [5]. Due to rapidly changing market requirements, factories have shifted from a centralized to a more decentralized structure in many areas of decision making, including scheduling. Since limited resources make scheduling an important decision in the production, accurate time synchronization and efficient scheduling solutions are vital for improving the productivity in a multi-factory production network [7].

Another relevant example is the case of Digital Twins (DTs) in Industry 4.0 [8]: the underlying idea is that a real product and its virtual counterpart are twins that travel a parallel journey from design and development to production and service life. A DT must be always in sync with the corresponding physical asset. Such synchronization is needed to correctly update and to keep the operational data (*i.e.*, failure and erroneous data) consistent within the production system.

Nowadays, several Industry 4.0 applications already imply stringent synchronization requirements (*e.g.*, sub-microsecond accuracy). Among the others, we consider the following ones as especially relevant:

- *Automation and control systems* [9], that need an accurate common notion of time as well as a reliable shared communication medium for timely data exchange, for instance to synchronize multi axis drive systems and subsystems with cyclic operation;
- *Measurement and automatic test systems* [10], that usually take advantage of accurate time stamping of logged data, for example to correlate acquired values in decentralized locations;
- *Power generation, transmission and distribution systems* [11,12], requiring a precise time synchronization of all the critical points within the power grid to accurately measure the delivered/consumed power and to predict critical load situations.

In this context, satellite-derived timing information plays a key role in the provisioning of an absolute time reference to a significant number of current and future connected systems. *Global Navigation Satellite System* (GNSS) receivers combined with specific transport protocols or with distributed synchronization approaches can indeed satisfy the stringent requirements foreseen in several industrial applications [13].

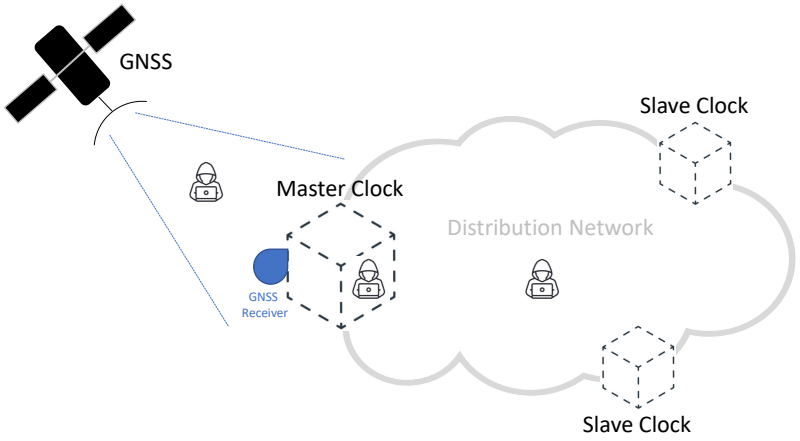


Figure 1. Simplified scheme of a GNSS-based synchronization distribution network.

Figure 1 provides a common scheme of a synchronization network, representative of a typical time distribution network and highlighting the role of GNSS technologies. In this scheme, accurate time information is generated by each GNSS satellite and then transmitted by means of properly formatted Radio-Frequency (RF) signals [14,15]. A Master Clock node of the network accurately synchronizes its local clock using a GNSS receiver, taking advantage of the signals received from the GNSS satellites visible at its location [16]. As illustrated in Figure 1, a distribution network allows to distribute the timing information to all the nodes, achieving an accurate synchronization between the Master Clock and multiple Slave Clocks. The distribution network can take advantage of different protocols and technologies, depending on the application requirements and constraints. Among the others, the *Precision Time Protocol* (PTP) represents a well-known and widely adopted solution for accurate time distribution in a network of clocks organized in a master-slave hierarchy [16,17]. Such synchronization protocol, also known as IEEE-1588 standard [18,19], allows for absolute time synchronization in the range of hundreds of nanoseconds through hardware assistance (SyncE) and, potentially, sub-nanosecond accuracy with the White Rabbit extension of PTP (WR-PTP) [20,21].

Nevertheless, a conscious adoption of the Industry 4.0 paradigm also requires the introduction of appropriate cybersecurity solutions. In fact, cyberattacks and hacking of factory industrial control systems are dramatically increasing in recent years. Proper mitigation measures against these attacks are needed to avoid the emergence of an

internet of *insecure* industrial things [22]. The same statement is also applicable to all the grids and telecommunications networks recognized as critical infrastructures [23,24], especially the upcoming 5G [16,17,25] that is considered as a key enabler for several Industry 4.0 applications. In all these cases, possible synchronization inaccuracies can directly result in a degradation of the quality of service provided by the communication network (*e.g.*, reduced throughput, increased latency and jitter) and, in extreme cases, in a complete disruption of the considered system or application.

Two areas in Figure 1 represent likely targets for potential attacks (*i.e.*, viable *attack vectors*) and, thus, need appropriate protection:

1. the **GNSS RF spectrum**, potentially targeted by attacks against the GNSS signals;
2. the **Time Distribution Network**, potentially targeted by attacks to the synchronization information. We can further divide this category into:
   (a) attacks targeting the Precision Time Protocol over the network, and
   (b) attacks against the integrity of PTP software (SW) running at each node and the related configuration files.

It is worth noting that recent scientific literature has extensively investigated attacks to the GNSS RF spectrum (*i.e.*, attack vector 1) and to the PTP protocol and packets over the network (*i.e.*, 2.a). Proper countermeasures against them are already available (*e.g.*, refer to [26–28] and references therein). For this reason, following sections will summarize these specific attack vectors, while their experimental assessment is out of the scope for this paper.

On the other hand, to the best of the authors' knowledge, previous works have not widely covered the attacks to the PTP SW integrity within nodes (*i.e.*, 2.b). This category of attacks is gaining special attention in both scientific and industrial communities, where the investigation of possible cybersecurity solutions is an active research topic, stressing the need for software integrity control at network nodes. For these reasons, we recognize this category of attacks and related solutions as especially relevant and challenging for Industry 4.0 applications and, thus, worth of further investigation.

Motivated by this background, the paper has the following objectives:

- an analysis of attack vectors for GNSS-based synchronization distribution networks, with a special attention to attacks not widely covered in prior work (*i.e.*, 2.b);
- the proposition of a novel solution for PTP nodes augmented with *Trusted Computing* technologies to counteract attacks to their integrity, complementing existing security solutions that focus on GNSS RF Spectrum and PTP protocol only;
- a reference implementation on a testbed, capable to demonstrate the proposed solution by means of quantitative results.

Our proposed approach differs from other *authentication and authorization-based* solutions already available in prior work and suitable to ensure the integrity of the network protocols and all type of exchanged packets. In fact, our solution leverages the Trusted Computing principles to achieve a level of trust in the behaviour of the synchronization distribution network. It is capable to ensure the integrity of the software and configuration of all the nodes and, thus, to avoid the exchange of incorrect synchronization information over secure protocols. For a more complete view on prior works about SW vulnerabilities in other domains, interested readers can also refer to [29–31] and the references therein.

In this sense, the contribution of the paper is twofold:

- We present our proposed implementation based on Trusted Computing and capable of protecting the integrity of the nodes of a synchronization distribution network.
- Next, we propose and describe a new testbed, suitable to emulate the identified attacks and to demonstrate the effectiveness of the proposed solution.

After this introduction, the paper is organized as follows. Section 2 provides an overview of the vulnerabilities and solutions related to the GNSS RF spectrum, while Section 3 covers the network attack vector. After that, Section 4 discusses the Trusted

Computing paradigm, as a suitable solution to protect the SW integrity of the nodes. Then, Section 5 describes in detail our implementation and Section 6 presents and discusses the obtained results. Finally, Section 7 concludes the paper with the main remarks.

2. GNSS RF Spectrum Attack Vector

GNSS technologies [14,15] have shown a remarkable growth in the last years, being nowadays adopted in the most different fields of applications such as: consumer devices (*e.g.*, smartphones, cameras, wearable devices, etc.), vehicular applications (*e.g.*, road user charging, bike sharing, connected and automated driving, etc.), manned and unmanned aviation (*e.g.*, drones), industrial applications and even in critical infrastructures. Every system that makes use of GNSS-derived data must trust them before taking decisions, especially in case of safety-critical or liability-critical applications.

The widespread adoption of GNSS technologies creates incentives for the attackers that want to impair or fool any system that rely on GNSS to estimate the position, the velocity and, especially, the *time* information [26]. GNSS receivers and connected devices integrating these receivers are all vulnerable to intentional attacks, aiming to affect the availability and the reliability of the GNSS signals and data. In general, GNSS RF attacks are classified in three main categories [24,26,27]:

1. **Jamming**, that is the blocking of the reception of GNSS signals by intentionally emitting RF interferences to disrupt the functionalities of the receivers, in order to reduce the signal-to-noise power level;
2. **Meaconing**, that corresponds to the rebroadcasting of delayed GNSS signals, without any distinction between signals received from different satellites;
3. **Spoofing**, that refers to the transmission of counterfeit GNSS-like signals, with the intent to produce false position and/or time data at the victim receiver.

Each of these three categories can be put in place in different ways, implying a different complexity and cost at the attacker side. Among the possible attacks, only few can be considered likely to be deployed in real applications [12,16,24].

Different countermeasures against these attacks are already available and widely discussed by the GNSS research community [26]. Please note that a comprehensive review of the state of the art related to GNSS RF attacks and proposed solutions is out of the scope for this paper. Nonetheless, interested readers can refer to [17,26,27] and the references therein.

Among the most recent solutions, it is worth to point out the on-going effort of the European Galileo program to gradually add authentication services to its first and second generation of satellites signals, in order to enable authentication functionalities for future civil receivers [27]. The Galileo *Open Service Navigation Message Authentication* (OSNMA) consists in a mechanism that allows the receivers to verify the authenticity of GNSS information, making sure that the data they receive are indeed from Galileo satellites and have not been modified in any way [32]. GNSS receivers can take advantage of such capability for implementing simple but effective detection methods against several spoofing attacks [16,24,33]. The first-ever signal-in-space transmission of Galileo OSNMA started on November 2020 and tests are currently underway in order to consolidate the service [34].

3. Time Distribution Network Attack Vector

The time distribution network represents a potential target for cyberattacks that can have an interest on degrading or disrupting the availability, integrity and reliability of the exchanged timing information.

In fact, several security mechanisms typically have a critical interdependence with timing synchronization. For instance, security solutions based on signed keys or certificates require accurate timing to determine whether they are valid and for how long. In this sense, accurate time synchronization is needed to establish a valid security system

and, on the other hand, a valid security system is required to confirm the accuracy of the timestamps [28]. The following paragraphs discuss the network attack vector, focusing on two sub-categories: (*i*) attacks against the PTP protocol used on the distribution network, and (*ii*) attacks against the PTP SW integrity and PTP configuration files of each node of the network.

3.1. Attacks Against the PTP Protocol and Packets

Recent scientific literature report several documented examples of attacks against timing protocols, including attacks against PTP [28,35,36]. These attacks are usually categorized as Denial-of-Service (DoS) attacks, feasible at various network layers, and Man-In-The-Middle (MITM), including clock masquerade, replay and filtering attacks [28,35,36]. For instance, Alghamdi and Schukat [37] proposed specific attack strategies and implementations: packet content manipulation, packet removal, packet delay manipulation, time source degradation, master spoofing, slave spoofing, compromised Best Master Clock Algorithm (BMCA), packet replay, and DoS. In addition, DeCusatis *et al.* [28] recently proposed and experimentally demonstrated a novel class of insider threats, including two variants of DoS spamming attacks, capable to incorrectly steer or permanently skew the slave's clock, and a master clock takeover attack.

Mitigation techniques and solutions against the previous categories of attacks have also been proposed and discussed. It is worth to mention that IEEE-1588-2008 standard [18] defined an experimental security extension (Annex K) in order to protect a PTP network. However, a number of weaknesses and drawbacks in this approach have been identified and, today, Annex K is deprecated [35]. Further on, a new version of the standard IEEE-1588-2019 has recently been published. It contains a new security extension in the Annex P, based on a multi-pronged approach [19]. Apart this remarkable standardization effort, the scientific literature includes other recent solutions. Among the others, it is worth to point out a proposed extension for PTP of the key management mechanism used in Network Time Security (NTS) [38] and an identity-based authentication system, *i.e.*, First Packet Authentication with Transport Access Control [28].

3.2. Attacks Against the SW Integrity and Configuration of the Nodes

The protection of the timing information over the GNSS RF spectrum and in transit over a PTP network against cyberattacks is not enough to ensure trust and rely on the GNSS precise timing. In fact, even assuming a state-of-the-art distribution network as in Figure 1, including all the previously cited security solutions, such system would be still vulnerable to potential attacks against the software integrity of Master and/or Slave nodes. Specific attacks targeting the integrity of the SW stack running within those nodes must also be considered to avoid that nodes exchange *incorrect synchronization information over secure protocols*.

Figure 2 illustrates a possible SW stack running on Master Clock and Slave Clock nodes. We adopt the simplified scheme in Figure 2 as a study case, intended to be as general as possible and representative of the typical SW modules installed and configured on real PTP nodes, but without the ambition to cover all the possible SW configurations and different hardware components (*e.g.*, Network Interface Card with or without PTP hardware timestamping, internal clock steering done via SW or with a dedicated FPGA, etc.).

We propose two possible configurations for the Master node on the left side and on the central part of Figure 2, respectively. Both these configurations include a GNSS module capable to receive the RF signals from one or multiple satellite constellations (*e.g.*, GPS, Galileo, GLONASS, and/or BeiDou) in order to estimate Position, Velocity, and Time (PVT) information. The GNSS receiver typically provides two outputs:

1. a textual interface over a serial communication protocol, providing the PVT data coded as "sentences" according to the National Marine Electronics Association (NMEA) 0183 standard [39], and

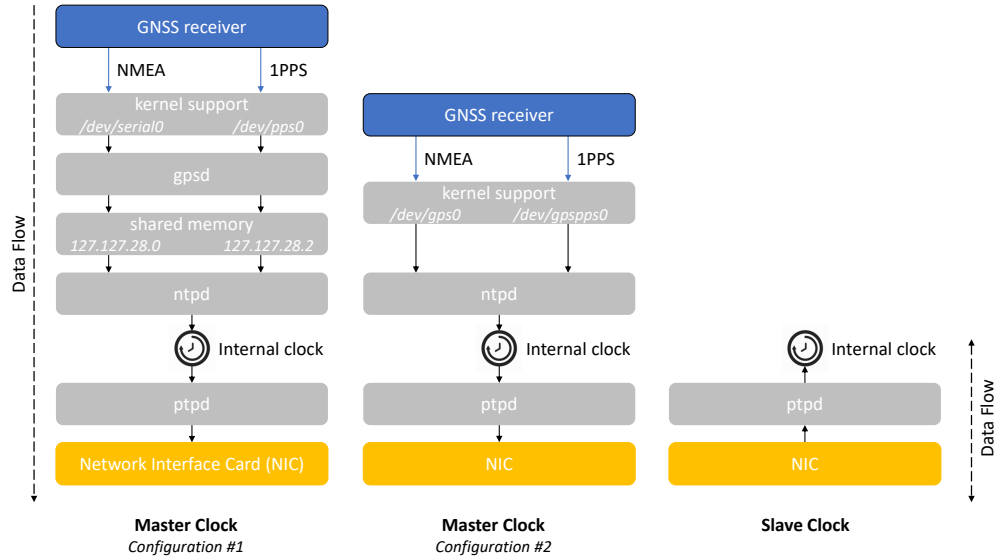


Figure 2. Possible SW stack and configurations for Master Clock and Slave Clock nodes.

2. the 1 Pulse-Per-Second (1PPS), that is a high precision analog signal having leading pulse edges synchronous with the beginning of each second of the time scale.

Different SW tools and/or daemons can accept as inputs the NMEA and 1PPS, taking advantage of the Linux kernel support to make them visible as devices [40].

In the configuration #1, these devices are a serial port for NMEA (`/dev/serial0`) and the Linux Pulse Per Second Application Programming Interface (PPSAPI) for 1PPS (`/dev/pps0`) [40]. A service daemon capable to parse the GNSS data (*i.e.*, `gpsd` [41]) uses both of them as inputs. We also shared such inputs to `ntpd`, a daemon implementing the Network Time Protocol (NTP) version 4 [42], by means of two shared memory segments (*i.e.*, `127.127.28.0` and `127.127.28.2`) properly configured using the Shared Memory Driver of `ntpd` [43]. In this way, `ntpd` is capable to correctly discipline the internal clock of the Master node, ensuring its accurate synchronization with respect to the GNSS time scale.

The configuration #2 represents a simplified alternative without `gpsd`. In this case, we directly provide the NMEA and 1PPS outputs of the GNSS receiver to `ntpd` by means of two symbolic links to the actual devices (*i.e.*, `/dev/gps0` and `/dev/gpspps0`, respectively). It is worth to notice that `ntpd` can use these symbolic links as two separate inputs, taking advantage of two dedicated drivers (*i.e.*, the Generic NMEA GPS Receiver [44] and the PPS Clock Discipline [45]) or as a single NMEA stream directly disciplined by the 1PPS (*i.e.*, by properly setting the `fudge flag1` of the Generic NMEA driver to enable the PPS signal processing [44]). The second option is preferable in order to simplify the configuration of `ntpd`, thus we will use it in the following discussion.

Once `ntpd` has correctly synchronized the internal clock of the Master node, both the configurations #1 and #2 adopt the PTP daemon (*i.e.*, `ptpd` [46]) to distribute the time synchronization over the network. In accordance to PTP working principles, `ptpd` can accurately synchronize multiple Slave nodes to the internal clock of the Master node. In detail, all the Slave nodes reach the `PTP_SLAVE` status [18,19], acquire the control and start correcting their internal clocks. It is known that `ptpd` can achieve microsecond level of time coordination, even on limited platforms [46].

The protection of the integrity of the SW stack in Figure 2 is of paramount importance to avoid that nodes share intentionally modified timing information over any secure version of PTP. Aiming to detect possible intentional changes in the work logic of the Master and/or the Slave node, next section proposes a software integrity architecture based on Trusted Computing principles and customized for embedded devices.

4. Trusted Computing

The Trusted Computing paradigm can be implemented in different ways and as a combination of different approaches and trust decisions to build the Secure Boot and/or Remote Attestation. The following paragraphs present our implementation for the purpose of protecting the integrity of an embedded system operating as a node of a synchronization distribution network.

4.1. Trusted Computing Base

Trusted Computing (TC) is used in this paper as per the Trusted Computing Group's (TCG) definition: a set of interoperable technologies to achieve a level of trust in the behaviour of an embedded system. The core element of TC is the hardware Root of Trust called Trusted Platform Module (TPM). A TPM is a tamper resistant piece of cryptographic hardware integrated with the system board that implements primitive cryptographic functions on top of which more complex features can be built in accordance with TCG specification TPM 2.0 [47]. The final objective for the application of TC in this work is to enable nodes to measure and prove cryptographically their integrity, *i.e.*, prove that the software running is the intended one and it has not been tampered with, to the other nodes involved in the synchronization distribution. Integrity measurement is therefore the process the nodes adopt to collect and digest the information about the integrity of their software and configuration for the purpose of being attested/verified. The three main hardware Roots of Trust to make the node a Trusted Computing Base (TCB) are:

1. Root of Trust for Measurements (RTM): the Core Root of Trust for Measurements (CRTM) that act as the Trust Anchor; it is commonly implemented by the initial bootloader secure code executed at power up or hardware reset;
2. Root of Trust for Storage (RTS): the TPM that provides (*i*) a set of Platform Configuration Registers (PCRs) to securely accumulate the integrity measurements in form of hashes, (*ii*) a key hierarchy architecture to securely store objects protected via encryption by the TPM on the file system;
3. Root of Trust for Reporting (RTR): the TPM that signs the values of a PCR set using an Attestation Key bound to the Endorsement Key, *i.e.*, a resident key that represents the TPM identity and that guarantees the origin and the integrity of the PCR values shared for the purpose of attestation.

4.2. Chain of Trust

The identification of the software components is performed through a Chain of Trust, *i.e.*, a hierarchy of trust rooted into a Trust Anchor (TA). The TA is an authoritative entity for which trust is assumed and not derived, *i.e.*, usually a small and carefully designed piece of firmware, executed by the system at power up or hardware reset. The TA is responsible for starting to build the Chain of Trust during bootstrap of the node. Each element of the chain is a software component that when executed performs its tasks and then identifies, loads and finally executes the next software component (see Figure 3).

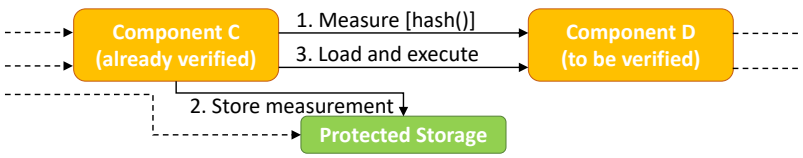


Figure 3. Basic Element of a Chain of Trust.

The identification of a component is made of two steps: the measurement, *i.e.*, a digest, calculated over that component and the storage of such measurement in the TPM, *i.e.*, the accumulation of the measurement via PCR extension operation.

Each PCR can contain one digest value which is the result of one or more PCR extension operations: $PCR_{i+1} = PCR_i||hash(component)$. For instance, the hash function can be `sha1`, `sha256`, or `sha384`. The Infineon TPM used in our prototype [48] implements `sha1` and `sha256` thus, for the sake of simplicity, we selected `sha1`. It is not possible to store a value directly in a PCR, but the only way to access a PCR for writing is through the PCR extend operation: this guarantees that all measurements are accumulated. By building on the fact that the Trust Anchor is trusted by definition and each software component is considered trusted at least to identify the next component, the identification of all components is trusted as well due to the transitive property of trust. The bootstrap process enhanced to build the Chain of Trust, *i.e.*, the capability of each software component to identify and load the next one, is called Authenticated Boot(strap). The Authenticated Boot goes from the CRTM up to the applications in userspace through bootloader and Linux kernel.

4.3. Trust Decision

The Trust Decision is about deciding whether a platform can be considered trusted for an intended purpose or not. It builds upon the Authenticated Boot, can occur (at any meaningful time) during it of after it and implies the cryptographic verification of the loaded software components to identify them and verify that they are the expected ones. If the Trust Decision is taken during the bootstrap, the verification is performed by the platform itself (usually using the TPM) over the components loaded until the decision time: based on the result of the verification, the bootstrap process can be stopped. This portion of the Authenticated Boot(strap) is called Secure Boot(strap). If the verification is performed after the bootstrap completion, it is called Remote Attestation, as it requires a remote entity, the Attestor, that performs the verification with the support of the trusted component on the platform, *i.e.*, the TPM. In our prototype we implemented both the Secure Boot and the Remote Attestation.

4.4. Secure Boot

The Secure Boot portion can be implemented by means of a combination of the (full) disk encryption with the TPM sealing of the encryption/decryption key to a specific set and configuration of the components loaded and executed before mounting the encrypted partition. Therefore, the availability of the encryption/decryption key is bound to specific values of a set of PCRs containing the accumulated measurements of the loaded components. If such components (and/or their configurations) are different from the expected ones, this is reflected into the PCR values different from those stored along with the encryption/decryption key in a Non-Volatile (NV) storage index. Indeed it is possible to associate a set of PCRs and their values representing a specific configuration to the disk encryption/decryption key when the data is protected (sealing). The complementary operation is the unsealing when the TPM checks whether the current PCR values meet the values' set stored along with the disk encryption/decryption key. If they differ, the TPM does not release the encryption/decryption key (it does the Trust Decision) and, since the encrypted partition cannot be mounted, the bootstrap process gets stopped, thus implementing the Secure Boot. In our prototype we implemented the Secure Boot using a modified version of Cryptsetup [49] that uses a NV storage index to protect the encryption/decryption phase.

4.5. Integrity Measurement Architecture (IMA)

Integrity Measurement Architecture (IMA) [50,51] is a security subsystem of the Linux kernel. When enabled, it measures all files that match a given policy (see Appendix A.1) that are loaded and possibly executed. All measurement records are kept in memory in the so-called IMA log and an overall integrity checksum is retained using the TPM PCR10. This is an excerpt of an IMA log for the Linux kernel v.4.19 we used:

```
PCR template-hash                              tpl filedata-hash                              file-pathname
10  03eb317e687cbd21e360e257b4810f8ac711fb4c ima 9797edf8d0eed36b1cf92547816051c8af4e45ee boot_aggregate
10  1110984f14fc70c87f90053612c3feaa068f66d6 ima 339c9d9d10d1fc25731d6f3d00b80e173e35456c /lib/systemd/systemd
10  bc447ab6e2f476455644dbcf9187c922418f02ba ima 369c4027e9b09131c6971ee963bfd37d41dc251c /lib/arm-linux-gnueabihf/ld-2.28.so
10  e359bd4b3a5b64f125dda645958237a123fe9fe7 ima 9e7916b2b9232f7db4cff863a6588e10253c0285 /etc/ld.so.preload
...
```

Each measurement record includes:

`IMA-record = PCR-index || template-hash || tpl || filedata-hash || file-pathname,`

where:

- `PCR-index` is the index of the used PCR, with default value equal to 10, *i.e.*, the PCR extended, one-by-one, by `template-hash`;
- `tpl` is the template name;
- `template-hash = sha1( filedata-hash length, filedata-hash, ... file-pathname length, file-pathname );`
- `filedata-hash = sha1( filedata ).`

IMA also includes the function *Appraisal* that together with the *Linux Extended Validation Module* (EVM) implements a fine grained Secure Boot process. In our prototype, to implement the Secure Boot we selected the approach based on an encrypted partition and the encryption/decryption key sealed to the TPM and a set of PCR values.

4.6. Remote Attestation

The Trust Decision can also occur when the authenticated bootstrap is completed; with this option a remote node, *i.e.*, the *Attestor*, is responsible to challenging and verifying the trustworthiness of target nodes with a given periodicity, as depicted in Figure 4. The Attestor performs a Remote Attestation by asking the TCB to be attested

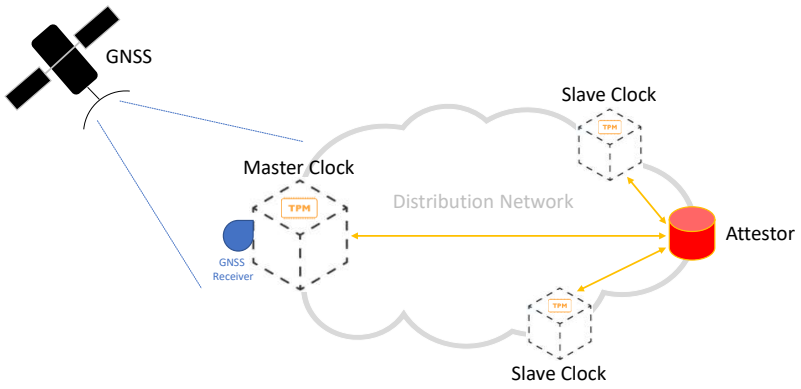


Figure 4. Illustration of Remote Attestation on the synchronization distribution network.

to send information about the loaded components for the cryptographic verification.

In this work the Remote Attestation is implemented in its easiest form. The Attestor opens a standard secure channel with mutual authentication by means of TLS [52] with the target TCB to protect the communication. Then, the Attestor challenges the TCB and verifies its trustworthiness. To avoid attacks where a malicious node can fool the Attestor by acting as MITM and acquire the attestation data from a trusted node, the node implements a soft binding between the TPM identity and the Secure Channel identity. This is implemented by binding the Attestation Key (AK) used to perform the *TPM_Quote* operation over the selected PCRs values and the key used for authentication during secure channel handshake. More practically, this is done by using the Public Key Certificate of the key used for authentication to extend one PCR of the TPM. The *TPM_Quote* is substantially a digital signature over a set of relevant PCR values and the nonce sent by the Attestor for freshness purposes to avoid replay attacks.

Once the TLS channel is established between the two parties that successfully identified each other, the Remote Attestation takes place over it according to the following sequence of actions:

1. The Attestor sends the nonce to the TCB,
2. the TCB executes the *TPM_Quote*, *i.e.*, it signs the PCRs from 0 to 10 and the nonce using the Attestation Key (AK), and prepares the IMA log for delivery,

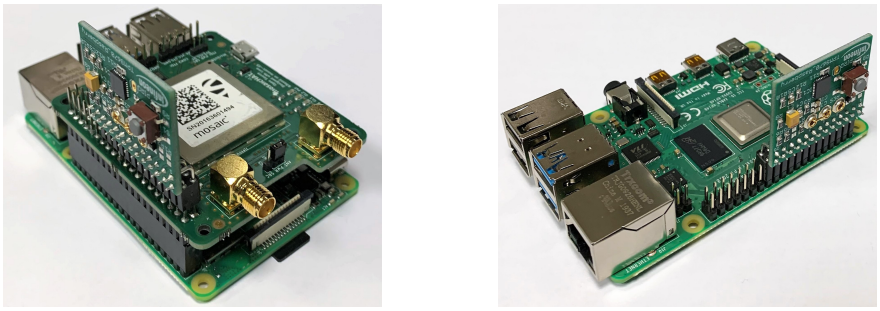
3. the Platform sends back to the Attestor the Quote (*i.e.,* the signature over the values of the PCR from 0 to 10 and the nonce) and the AK certificate, the PCR values, the IMA log, and the list of files measured by the bootloader with their digest; then,
4. the Attestor performs all checks to verify the TCB integrity:
   (a) it verifies the validity of the Quote using the public key embedded in the AK certificate (previously set as trusted), also including the nonce;
   (b) it verifies that the values of "soft PCRs" calculated from the list of files (and their digest) measured by the bootloader correspond to the values of PCR 0 to 8 extended as well by the bootloader;
   (c) it calculates the digest of the AK certificate and the value of a "soft PCR" and verifies that corresponds to PCR9 value;
   (d) it verifies the integrity of the IMA log by recalculating the value of a "soft PCR" from the IMA log itself and comparing it with the value of PCR10;
   (e) it verifies the files included in the IMA log with the exception of the files to be excluded, (*e.g.,* the log files) against whitelists/blacklists of digests previously built and exchanged in a trusted manner. The result of the integrity status can be: `trusted` if all files in the IMA log are found in whitelists; `unknown` if at least one file in the IMA log is not found in whitelists but no file is found in blacklists; and `untrusted` if at least one file in the IMA log is found in blacklists or if any of the checks from (a) to (d) fails.
5. Finally, the Attestor closes the TLS channel and takes the appropriate action(s).

Which are the appropriate action(s) depends on the specific Industry 4.0 application and they are subject to further research.

5. Reference Implementation

The following paragraphs provide a description of our proposed testbed, intended to be representative of a typical synchronization distribution network. We have implemented the nodes of the testbed by means of the following components:

- Raspberry Pi® 4 (RPi4) Model B [53], a flexible and high-performance Single Board Computer with a well-supported set of SW libraries and tools for the Linux environment;
- mosaicHAT [54], an open source hat compatible with RPi4. It is based on the Septentrio's mosaic-X5® receiver [55], a multi-band, multi-constellation GNSS module representative of the state of the art and supporting the Galileo signals (*i.e.,* OSNMA-ready);
- Infineon OPTIGA™ TPM SLI 9670 Iridium board [48], an evaluation board with the widely used TPM2.0 chip.



(**a**) (**b**)

Figure 5. Picture of the Master clock node (**a**) and Slave clock node (**b**).

In detail, Figure 5 (a) shows a picture of a Master node in our testbed, consisting in a RPi4 with both the mosaicHAT and the TPM stacked on top of it, while Figure 5 (b) presents a Slave node (*i.e.,* a RPi4 with the TPM only).

These components are instrumental to build a highly flexible and reconfigurable testbed, thus suitable to analyse relevant attack vectors and countermeasures in a controlled and legal framework. Our testbed is also scalable, thus we can easily extend it to emulate different network topologies, potentially including a large number of nodes, but avoiding the complexities related to the operation of real PTP hardware in a synchronization distribution network.

For this purpose, Figure 6 shows a network topology based on five nodes that we will adopt in all the tests reported in next section.

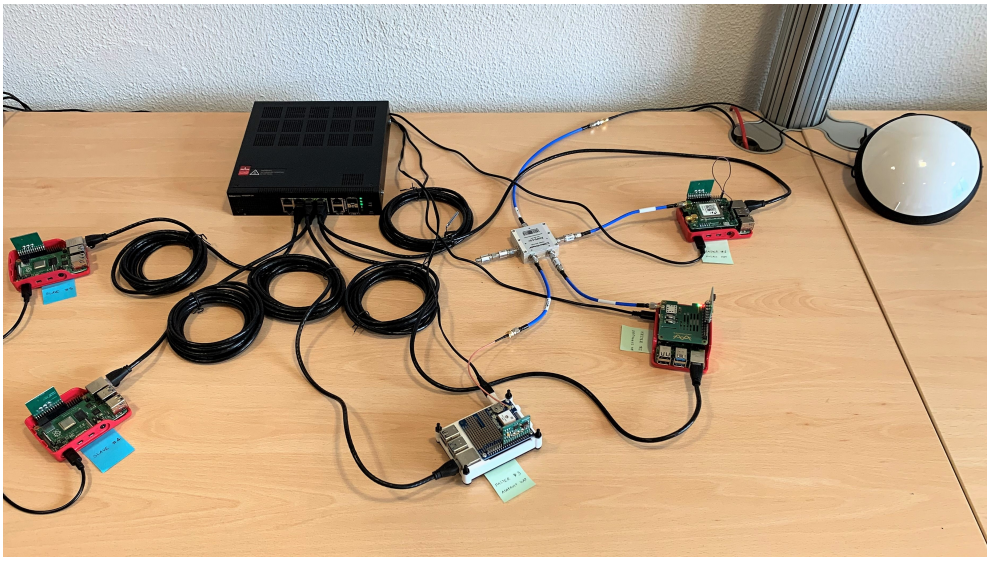


Figure 6. Picture of the full testbed, configured with three Master nodes and two Slave nodes.

The picture presents the following components, clockwise starting from the upper right corner:

- a professional GNSS antenna (*i.e.*, Tallysman® VSP6037L VeroStar™ Full GNSS Precision Antenna plus L-band [56]), placed on the desk for illustration purposes only, but properly mounted on the rooftop of the building during the tests;
- three RPi4 configured as Master nodes (*i.e.*, Master 1, 2, and 3), including both a GNSS module and a TPM stacked on top of it;
- two RPi4 configured as Slave nodes (*i.e.*, Slave 4 and 5), without a GNSS module.

All the five RPi4 nodes in Figure 6 are connected to the power supply and to the same Local Area Network (LAN) by means of a switch and Ethernet cables. As far as the three Master nodes are concerned, we connected them to the same GNSS antenna through a signal splitter. Only the Master 1 is equipped with the mosaicHAT [54], while the other two Master nodes have low cost GNSS modules (*i.e.*, Uputronics™ Raspberry Pi GPS/RTC Expansion Board [57] for Master 2, Adafruit Ultimate GPS HAT [58] for Master 3). Moreover, the Master 1 is configured as the Grand Master clock for the PTP protocol, while the Master 2 acts as backup master clock and the Master 3 (*i.e.*, the white node in the middle of Figure 6) is used as a monitoring node. In detail, the Master 3 is capable to estimate the relative synchronization errors of the other nodes on the LAN in real time, taking advantage of the `ntpq` utility [59], and to collect detailed log files, by means of a properly configured `ntpd` daemon [42].

It must be remarked that this setup is potentially suitable to a comparative performance assessment of the different GNSS modules and/or for emulating different attacks against the GNSS RF Spectrum (see Section 2). Nonetheless, as previously highlighted in Section 1, GNSS RF vulnerabilities are out of the scope for this paper, thus the role of the GNSS modules in our testbed is just to provide a reliable time synchronization to all the RPi4 nodes. For these reasons, the experiments presented in following section will focus only on attacks against the SW integrity and configuration of the nodes.

6. Results and Discussion

The testbed introduced in previous section is suitable to test different SW stacks and configurations and to comparatively assess them in terms of performance and robustness. For instance, we can test both the configurations #1 and #2 as in previous Figure 2 on the Master node. In the first case, the `gpsmon` utility program [60] allows to check the behaviour of `gpsd` in real-time [41]. On the other hand, the `ntpq` utility [59] is usable in both configurations to monitor the status of `ntpd` [42] and then to estimate the synchronization performance of the Master 1 in our testbed.

It is worth to recall that our testbed adopts the `ntpd` daemon to accurately synchronize the local clock of the Master node to the GNSS time scale, while `ptpd` distributes the timing information from the Master to multiple Slave nodes. Typical time distribution networks can require a remarkable amount of time (*e.g.*, ranging from few minutes up to several days) to achieve an initial synchronization between multiple nodes. In fact, small frequency and phase corrections allow to gradually and continuously adjust (*i.e.*, *discipline*) all the clocks. The required amount of time for these operations mainly depends on the quality of the local oscillators of the nodes, on the specific configuration of `ntpd` and `ptpd`, and on the application requirements in terms of synchronization accuracy and precision.

As an example, Table 1 summarizes the obtained results running both the previous configurations #1 and #2 for different time intervals (*i.e.*, from 5 minutes up to 3 days). In detail, Table 1 reports offset and jitter values (in milliseconds) estimated by `ntpq` for both 1PPS and NMEA outputs in configuration #1, while in configuration #2 such values are available just for 1PPS.

Table 1. Obtained time offset and jitter values (in ms) in different configurations

Obtained outputs		1PPS		NMEA	
Duration	Configuration	offset	jitter	offset	jitter
5 minutes	#1	-4.224	1.106	-4.477	1.147
	#2	-3.636	0.896	–	–
10 minutes	#1	0.038	0.022	-0.306	0.135
	#2	0.032	0.018	–	–
1 hour	#1	0.001	0.001	0.190	0.130
	#2	-0.001	0.001	–	–
3 days	#1	0.001	0.001	1.287	0.182
	#2	-0.001	0.001	–	–

The 1PPS columns demonstrate that the configurations #1 and #2 result in equivalent performance after 1 hour (*i.e.*, a synchronization accuracy of the order of 1 μs). Thus, we select the configuration #2 in order to simplify the SW stack and to reduce possible vulnerabilities in the Master nodes and we will use it in the following tests.

The testbed also provides a playground to test different attacks. For instance, the network topology shown in Figure 6 is suitable to emulate specific attacks against the PTP SW integrity of the nodes, according to the previous discussion in Section 3.2. In detail, Figure 7 reports the obtained results considering the following attacks:

1. an attack against the SW integrity of the `ptpd` daemon;
2. an attack against the `ptpd` configuration file.

The Attack 1 emulates an attacker that has gained access with superuser privileges to the two Slave nodes. The attack consists in the installation of a maliciously modified executable file on both the nodes, obtained by modifying the original source code of `ptpd` [46]. In our test, the modified SW introduces an instantaneous time bias on all the time stamps received by each Slave node. We insert such bias in the lowest possible level of the `ptpd` source code, as close as possible to the NIC hardware (*i.e.*, by adding 12 lines of code to the `net.c` source file). The instantaneous bias has an initial small magnitude,

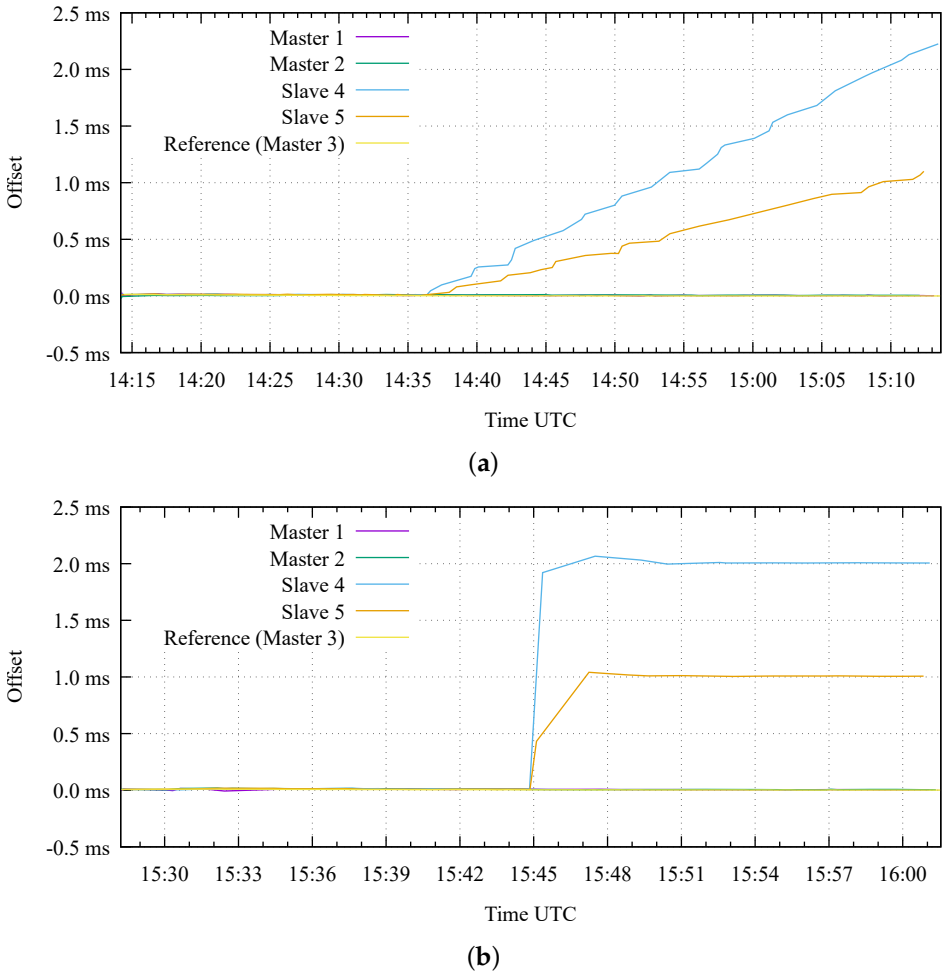


Figure 7. Obtained results on the testbed during the first attack against the SW integrity (**a**) and the second attack against the configuration of the PTP daemon (**b**).

intended to be undetectable with conventional measures (*e.g.*, on `ptpd` statistics and event logs), and it gradually increases with a consistent sign over the duration of the attack, thus resulting in a frequency drift. In this way, a sufficiently long attack duration can produce arbitrarily large time offsets on the target nodes, potentially with different relative time offsets (i.e., different error magnitude and/or sign). As shown in Figure 7(a), the first test lasts for approximately one hour and begins with an initial phase in nominal conditions, where all the five nodes are correctly synchronized. Then, the attack actually starts to introduce a time bias at 14:36. In this test, we intentionally introduce a different frequency drift in the two victim nodes (*i.e.*, 1 μs/s on Slave 4 and 0.5 μs/s on Slave 5) during the second part of the test. For this reason, the Slave 4 reaches a time synchronization error larger than 2 ms at the end of the test, while the Slave 5 shows an error larger than 1 ms with respect to all the other Master nodes.

On the other hand, the Attack 2 emulates a simpler scenario where we assume that the attacker is only capable to edit the `ptpd` configuration file (and not its executable file). In practice, the modified configuration file on each target Slave node introduces a constant time bias with an arbitrarily large magnitude on all the time stamps received from the Master node. Such large bias can result in a time step on the victim clocks during the attack execution, potentially detectable with conventional measures (*e.g.*, on `ptpd` statistics and event logs). We emulate this attack on our testbed by setting a wrong value for one of the parameters in the `ptpd` configuration file (*i.e.*, `ptpengine:offset_shift`).

In this way, we introduce an error of 2 ms and 1 ms for Slave 4 and Slave 5, respectively, thus resulting in different relative time offsets, as demonstrated in Figure 7(b).

Trusted Computing principles enables the mitigation of both these attacks. In fact, the Remote Attestation is capable to detect the unauthorized replacement of an executable file (or a shared library), when the Attestor compares the IMA Log with whitelists/blacklists. The same consideration applies for an unauthorized modification to a configuration file.

In the following example we first measure as good the executable of the `ptpd` daemon and its configuration file, as the measurements in the IMA log coincide with the ones in whitelist. An excerpt of the output of the command showing the IMA log is as follows:

```
pi@raspberrypi:~/$ sudo cat /sys/kernel/security/ima/ascii_runtime_measurements
10 03eb317e687cbd21e360e257b4810f8ac711fb4c ima 9797edf8d0eed36b1cf92547816051c8af4e45ee boot_aggregate
10 1110984f14fc70c87f90053612c3feaa068f66d6 ima 339c9d9d10d1fc25731d6f3d00b80e173e35456c /lib/systemd/systemd
10 bc447ab6e2f476455644dbcf9187c922418f02ba ima 369c4027e9b09131c6971ee963bfd37d41dc251c /lib/arm-linux-gnueabihf/ld-2.28.so
10 e359bd4b3a5b64f125dda645958237a123fe9fe7 ima 9e7916b2b9232f7db4cff863a6588e10253c0285 /etc/ld.so.preload
...
10 26bf9166f38fd64c452484ddc48d91c32913b53b ima 77a352d6c2817ef4c6df25512f104e46fcf1d6e0 /usr/local/sbin/ptpd2
10 d52997919ea0fe1cdec53bf2546d3db3076e9f58 ima b526bd78d18255929e2c2d2ccd821c47abc10912 /home/pi/PTPd/ptpd2.slave.conf
...
```

The last two rows are related to the `ptpd` executable file and to its configuration file installed in the Slave 4 and Slave 5 nodes, including their full pathnames.

An excerpt of the whitelist file is the following:

```
b526bd78d18255929e2c2d2ccd821c47abc10912  /home/pi/PTPd/ptpd2.slave.conf
77a352d6c2817ef4c6df25512f104e46fcf1d6e0  /usr/local/sbin/ptpd2
01a2ecb8982b43a7e3cd9fb9af2e239def49d073  /usr/local/sbin/tpm2-abrmd
bc595d77c2cea5eb899927d0a26d88292eaa64f5  /usr/local/bin/ima_boot_aggregate
fab6bd62f87f58395c37e470a9c1efd2d6d08f4c  /usr/local/bin/ima_measure
a93155873b9f2becbbab6e8641b5a2d566fd678f  /usr/local/bin/ima_mmap
20436ac9bbf6eeee3567bcd3fd08b4c845578df0  /usr/local/bin/ra_build_white_list
```

Looking at the first two rows of the whitelist, we can appreciate that the digest values for both the configuration file and the `ptpd` executable are consistent with the corresponding values of the previous IMA log.

At this point, we simulate the previous Attack 1 (*i.e.*, an unauthorized SW modification) by stopping the PTP daemon, replacing the `ptpd` executable with a maliciously modified version, and then running it with the nominal configuration, as follows:

```
pi@raspberrypi:~/$ sudo killall ptpd2
pi@raspberrypi:~/$ sudo cp ~/PTPdm/ptpd/src/ptpd2 /usr/local/sbin/ptpd2
pi@raspberrypi:~/$ sudo ptpd2 -c /home/pi/PTPd/ptpd2.slave.conf
```

These commands trigger new IMA measurements, which result in the following records appended to the IMA log:

```
10 2e8a62501ac82d1b51f073d019d5e6d41e5999cf ima e1299122fc1dfdb0707a96bf6b879273278c941a /usr/bin/killall
10 c7fbe7019734b0d664c0e0bc8e407923b50d9b03 ima 19e940258bfba6cc25e66c73bb0d7939d9583a1c /home/pi/PTPdm/ptpd/src/ptpd2
10 c57611f6a5a369a0b0f9412dd307f78defc8e725 ima 19e940258bfba6cc25e66c73bb0d7939d9583a1c /usr/local/sbin/ptpd2
```

By comparing the IMA log and the whitelist, the attack is immediately detected, as the digest for `ptpd` in the former is different from the new digest in the latter.

At the end of that test, we restore the original `ptpd` and reboot both the Slave nodes in order to return to the nominal synchronization of all the nodes. Then, we can emulate the Attack 2 (*i.e.*, an unauthorized modification on the configuration file). First we retrieve the Process ID (PID) of the running `ptpd` daemon by watching the content of its status log file, then we overwrite the original configuration file with a maliciously

modified version and, finally, we send the appropriate signal (*i.e.*, `SIGHUP`) to `ptpd` in order to force it to reload its configuration file:

```
pi@raspberrypi:~/$ watch -n 1 cat /var/log/ptpd2.status.log
pi@raspberrypi:~/$ sudo cp ~/PTPd/ptpd2.slave_offset.conf ~/PTPd/ptpd2.slave.conf
pi@raspberrypi:~/$ sudo kill -s SIGHUP <PID>
```

As in previous attack, these commands trigger new IMA measurements, as follows:

```
10 e78bbb133849e6ec45dd9e2dd2b87b13eaa8ba14 ima bffac84554ed0fc938387d163dae108d26f80341 /usr/bin/watch
10 06f51b1aee4b32d001ab3a4c905530de203812ea ima 3fd479d4a21b4c7a07df154156b69c8be7c2b5a1 /home/pi/PTPd/ptpd2.slave_offset.conf
10 fc49caa91dadee8af0540791f96b9f4f3483b56a ima 1ea5039520c120ab8c77e870c096e7b7d6908ee4 /bin/kill
10 d67b1554ac31101b987b1dd9537e8c45a659a705 ima 3fd479d4a21b4c7a07df154156b69c8be7c2b5a1 /home/pi/PTPd/ptpd2.slave.conf
```

In this case the attack is detectable by comparing the digest of the configuration file in the IMA log with the previous value stored in the whitelist.

It is worth to clarify that, for both attacks, the detection by the Attestor can have a potential delay with respect to the actual attack initiation. Such delay depends on the given periodicity at which the Attestor contacts the nodes to attest/verify the integrity of their software and configuration: a frequent execution of such Remote Attestation protocol allows a rapid detection of an ongoing attack, but it comes at the price of an increased computational load and network overhead. In this sense, the periodicity of the attestation procedure represents an important design parameter to be tailored to each use case in Industry 4.0 applications, trading-off security versus complexity.

7. Conclusions

This paper has provided a complete analysis of the possible attack vectors to a GNSS-based Time Distribution Network, shedding the light on the importance of deploying a software integrity architecture in Industry 4.0 applications. We have proposed and demonstrated a Trusted Computing implementation for embedded devices, together with a real-world option to make a flexible and scalable testbed for further exploration of threats and vulnerabilities due to the combination of diverse attack vectors.

Author Contributions: All authors have contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This work was developed within the ROOT project (www.gnss-root.eu), funded by the European GNSS Agency under the European Union's Horizon 2020 – G.A. n. 101004261.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No data available online. For further query please contact the corresponding author.

Acknowledgments: The authors would like to thank Gianluca Ramunno for the technical support provided during the initial part of this work, all the partners of the ROOT project for their useful suggestions and, finally, the reviewers whose comments helped to improve and clarify this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1 Default IMA policy

The default IMA policy for the measurement, called `ima_tcb`, is the following:

```
dont_measure fsmagic=PROC_SUPER_MAGIC
dont_measure fsmagic=SYSFS_MAGIC
dont_measure fsmagic=DEBUGFS_MAGIC
```

```
dont_measure fsmagic=TMPFS_MAGIC
dont_measure fsmagic=SECURITYFS_MAGIC
dont_measure fsmagic=SELINUX_MAGIC
measure func=BPRM_CHECK
measure func=FILE_MMAP mask=MAY_EXEC
measure func=PATH_CHECK mask=MAY_READ uid=0
```

and instructs IMA to measure all files loaded for execution and all files read under the user account `root`, with the exclusion of all special file systems. Files that get modified during OS run-time are re-measured by IMA and can appear multiple time in the IMA log with different measurements. In our prototype the variable files, like the log files located in `/var/log` are measured but excluded during the verification that takes place with the Remote Attestation.

References

1. Calia, E.; D'Aprile, D., Industry4.0. In *Analytics for the Sharing Economy: Mathematics, Engineering and Business Perspectives*; Crisostomi, E.; Ghaddar, B.; Häusler, F.; Naoum-Sawaya, J.; Russo, G.; Shorten, R., Eds.; Springer International Publishing: Cham, 2020; pp. 309–333. doi:10.1007/978-3-030-35032-1_18.
2. Morella, P.; Lambán, M.P.; Royo, J.A.; Sánchez, J.C. The Importance of Implementing Cyber Physical Systems to Acquire Real-Time Data and Indicators. *J* **2021**, *4*, 147–153. doi:10.3390/j4020012.
3. Aceto, G.; Persico, V.; Pescapé, A. A Survey on Information and Communication Technologies for Industry 4.0: State-of-the-Art, Taxonomies, Perspectives, and Challenges. *IEEE Communications Surveys Tutorials* **2019**, *21*, 3467–3501. doi:10.1109/COMST.2019.2938259.
4. Xu, H.; Yu, W.; Griffith, D.; Golmie, N. A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective. *IEEE Access* **2018**, *6*, 78238–78259. doi:10.1109/ACCESS.2018.2884906.
5. Raptis, T.P.; Passarella, A.; Conti, M. Data Management in Industry 4.0: State of the Art and Open Challenges. *IEEE Access* **2019**, *7*, 97052–97093. doi:10.1109/ACCESS.2019.2929296.
6. Puttnies, H.; Danielis, P.; Sharif, A.R.; Timmermann, D. Estimators for Time Synchronization—Survey, Analysis, and Outlook. *IoT* **2020**, *1*, 398–435. doi:10.3390/iot1020023.
7. Behnamian, J.; Fatemi Ghomi, S. A survey of multi-factory scheduling. *J. Intell. Manuf.* **2016**, *27*, 231–249. doi:10.1007/s10845-014-0890-y.
8. Sahal, R.; Alsamhi, S.H.; Breslin, J.G.; Brown, K.N.; Ali, M.I. Digital Twins Collaboration for Automatic Erratic Operational Data Detection in Industry 4.0. *Applied Sciences* **2021**, *11*. doi:10.3390/app11073186.
9. Kerö, N.; Puhm, A.; Kernen, T.; Mroczkowski, A. Performance and Reliability Aspects of Clock Synchronization Techniques for Industrial Automation. *Proceedings of the IEEE* **2019**, *107*, 1011–1026. doi:10.1109/JPROC.2019.2915972.
10. Li, M. Clock Synchronization Technology Research for Distributed Automatic Test System. Machine Tool Technology, Mechatronics and Information Engineering. Trans Tech Publications Ltd, 2014, Vol. 644, *Applied Mechanics and Materials*, pp. 891–894. doi:10.4028/www.scientific.net/AMM.644-650.891.
11. Delle Femine, A.; Gallo, D.; Landi, C.; Luiso, M. The Design of a Low Cost Phasor Measurement Unit. *Energies* **2019**, *12*. doi:10.3390/en12142648.
12. Pini, M.; Falletti, E.; Nicola, M.; Margaria, D.; Marucco, G. Dependancy of power grids to satellite-derived time: vulnerabilities and new protections. 2018 IEEE International Telecommunications Energy Conference (INTELEC), 2018, pp. 1–8. doi:10.1109/INTLEC.2018.8612407.
13. Petrov, D.; Melnik, S.; Hämäläinen, T. Distributed GNSS-based Time Synchronization and applications. 2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2016, pp. 130–134. doi:10.1109/ICUMT.2016.7765345.
14. Bacci, G.; Falletti, E.; Fernández-Prades, C.; Luise, M.; Margaria, D.; Zanier, F. Chapter 2 - Satellite-Based Navigation Systems. In *Satellite and Terrestrial Radio Positioning Techniques*; Dardari, D.; Falletti, E.; Luise, M., Eds.; Academic Press: Oxford, 2012; pp. 25–74. doi:10.1016/B978-0-12-382084-6.00002-7.
15. Dovis, F.; Margaria, D.; Mulassano, P.; Dominici, F., Overview of Global Positioning Systems. In *Handbook of Position Location*; John Wiley and Sons, Ltd, 2018; chapter 20, pp. 655–705. doi:10.1002/9781119434610.ch20.
16. Pini, M.; Minetto, A.; Vesco, A.; Berbecaru, D.; Contreras Murillo, L.M.; Nemry, P.; De Francesca, I.; Rat, B.; Callewaert, K. Satellite-derived Time for Enhanced Telecom Networks Synchronization: the ROOT Project. 2021 IEEE 8th International Workshop on Metrology for AeroSpace (MetroAeroSpace), 2021, pp. 288–293. doi:10.1109/MetroAeroSpace51421.2021.9511780.
17. Pini, M.; Minetto, A.; Nemry, P.; Rat, B.; Contreras Murillo, L.M.; De Francesca, I.; Margaria, D.; Vesco, A.; Berbecaru, D.; Callewaert, K.; Dovis, F.; Lioy, A. Protection of GNSS-based Synchronization in Communication Networks: The ROOT project. Submitted to the European Navigation Conference & International Navigation Conference (Navigation 2021), 2021.

18. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)* **2008**, pp. 1–300. doi:10.1109/IEEESTD.2008.4579760.
19. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2019 (Revision IEEE Std 1588-2008)* **2020**, pp. 1–499. doi:10.1109/IEEESTD.2020.9120376.
20. Girela-López, F.; López-Jiménez, J.; Jiménez-López, M.; Rodríguez, R.; Ros, E.; Díaz, J. IEEE 1588 High Accuracy Default Profile: Applications and Challenges. *IEEE Access* **2020**, *8*, 45211–45220. doi:10.1109/ACCESS.2020.2978337.
21. Lipiński, M.; Włostowski, T.; Serrano, J.; Alvarez, P. White rabbit: a PTP application for robust sub-nanosecond synchronization. 2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, 2011, pp. 25–30. doi:10.1109/ISPCS.2011.6070148.
22. Urquhart, L.; McAuley, D. Avoiding the internet of insecure industrial things. *Computer Law & Security Review* **2018**, *34*, 450–466. doi:10.1016/j.clsr.2017.12.004.
23. Council of the European Union, Brussels, Belgium. 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. [online], 2008. Available at https://eur-lex.europa.eu/eli/dir/2008/114/oj.
24. Falletti, E.; Margaria, D.; Marucco, G.; Motella, B.; Nicola, M.; Pini, M. Synchronization of Critical Infrastructures Dependent Upon GNSS: Current Vulnerabilities and Protection Provided by New Signals. *IEEE Systems Journal* **2019**, *13*, 2118–2129. doi:10.1109/JSYST.2018.2883752.
25. Ruffini, S.; Johansson, M.; Pohlman, B.; Sandgren, M. 5G synchronization requirements and solutions. [online], 2021. Available at https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-synchronization-requirements-and-solutions.
26. Dovis, F. *GNSS Interference Threats and Countermeasures*; Artech House: Norwood, MA, 2015; p. 216.
27. Margaria, D.; Motella, B.; Anghileri, M.; Floch, J.; Fernandez-Hernandez, I.; Paonni, M. Signal Structure-Based Authentication for Civil GNSSs: Recent Solutions and Perspectives. *IEEE Signal Processing Magazine* **2017**, *34*, 27–37. doi:10.1109/MSP.2017.2715898.
28. DeCusatis, C.; Lynch, R.M.; Kluge, W.; Houston, J.; Wojciak, P.A.; Guendert, S. Impact of Cyberattacks on Precision Time Protocol. *IEEE Transactions on Instrumentation and Measurement* **2020**, *69*, 2172–2181. doi:10.1109/TIM.2019.2918597.
29. Jurcut, A.D.; Ranaweera, P.; Xu, L., Introduction to IoT Security. In *IoT Security: Advances in Authentication*; 2020; pp. 27–64. doi:10.1002/9781119527978.ch2.
30. Nebbione, G.; Calzarossa, M.C. Security of IoT Application Layer Protocols: Challenges and Findings. *Future Internet* **2020**, *12*. doi:10.3390/fi12030055.
31. Harbi, Y.; Aliouat, Z.; Refoufi, A.; Harous, S. Recent Security Trends in Internet of Things: A Comprehensive Survey. *IEEE Access* **2021**, *9*, 113292–113314. doi:10.1109/ACCESS.2021.3103725.
32. Fernández-Hernández, I.; Rijmen, V.; Seco-Granados, G.; Simon, J.; Rodríguez, I.; Calle, J.D. A Navigation Message Authentication Proposal for the Galileo Open Service. *NAVIGATION* **2016**, *63*, 85–102. doi:10.1002/navi.125.
33. Margaria, D.; Marucco, G.; Nicola, M. A first-of-a-kind spoofing detection demonstrator exploiting future Galileo E1 OS authentication. 2016 IEEE/ION Position, Location and Navigation Symposium (PLANS), 2016, pp. 442–450. doi:10.1109/PLANS.2016.7479732.
34. European Union Agency for the Space Programme. Tests of Galileo OSNMA underway. [online], 2021. Available at https://www.euspa.europa.eu/newsroom/news/tests-galileo-osnma-underway.
35. Alghamd, W.; Schukat, M. A Detection Model Against Precision Time Protocol Attacks. 2020 3rd International Conference on Computer Applications Information Security (ICCAIS), 2020, pp. 1–3. doi:10.1109/ICCAIS48893.2020.9096742.
36. O'Donoghue, K. Emerging solutions for time protocol security. 2016 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), 2016, pp. 1–6. doi:10.1109/ISPCS.2016.7579502.
37. Alghamdi, W.; Schukat, M. Practical Implementation of Cybersecurity Attacks on PTP Networks. 2020 International Timing and Sync Forum (ITSF), 2020.
38. Arnold, D.; Langer, M. Adapting NTS to PTP. 2020 International Timing and Sync Forum (ITSF), 2020.
39. National Marine Electronics Association. NMEA 0183 Interface Standard, Version 4.11. [online], 2018. Available at https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard.
40. Hiranuma, K.; van Heusden, F. LinuxPPS wiki. [online], 2020. Available at http://linuxpps.org.
41. The GPSD project. gpsd(8) Manual Page. [online], 2021. Available at https://gpsd.io/gpsd.html.
42. The NTP (R&D) Project. ntpd - Network Time Protocol (NTP) Daemon. [online], 2014. Available at http://doc.ntp.org/current-stable/ntpd.html.
43. The NTP (R&D) Project. Shared Memory Driver. [online], 2014. Available at http://doc.ntp.org/current-stable/drivers/driver28.html.
44. The NTP (R&D) Project. Generic NMEA GPS Receiver Driver. [online], 2020. Available at http://doc.ntp.org/current-stable/drivers/driver20.html.
45. The NTP (R&D) Project. PPS Clock Discipline Driver. [online], 2014. Available at http://doc.ntp.org/current-stable/drivers/driver22.html.
46. Owczarek, W.; Kreuzer, S.; Neville-Neil, G.V. PTPd official source - Precision Time Protocol daemon (1588-2008). [online], 2019. Available at https://github.com/ptpd/ptpd.

47. Trusted Computing Group (TCG). Trusted Platform Module Library Specification, Family 2.0, Level 00, Revision 01.59. [online], 2019. Available at https://trustedcomputinggroup.org/resource/tpm-library-specification/.
48. Infineon Technologies AG. OPTIGA™ TPM Application Note. Integration of an OPTIGA™ TPM SLx 9670 TPM2.0 with SPI Interface in a Raspberry Pi® 4 Linux environment. [online], 2019. Available at https://www.infineon.com/dgdl/Infineon-OPTIGA_SLx_9670_TPM_2.0_Pi_4-ApplicationNotesv07_19-EN.pdf?fileId=5546d4626c1f3dc3016c3d19f43972eb.
49. Fuchs, A. Cryptsetup TPM Incubator. [online], 2019. Available at https://github.com/AndreasFuchsSIT/cryptsetup-tpm-incubator/tree/luks2tpm.
50. Sailer, R.; Zhang, X.; Jaeger, T.; van Doorn, L. Design and Implementation of a TCG-based Integrity Measurement Architecture. 13th USENIX Security Symposium (USENIX Security 04); , 2004.
51. Kasatkin, D.; Zohar, M. Integrity Measurement Architecture. [online], 2017. Available at https://sourceforge.net/p/linux-ima/wiki/Home/.
52. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446. [online], 2018. Available at https://www.rfc-editor.org/info/rfc8446.
53. Raspberry Pi® Trading Ltd. Raspberry Pi® 4 Computer Model B, Product brief. [online], 2021. Available at https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf.
54. Sa'd, J. MosaicHAT: an open source Raspberry Pi HAT based on Septentrio's mosaic-X5. [online], 2020. Available at https://github.com/septentrio-gnss/mosaicHAT.
55. Septentrio NV. mosaic-X5®: Compact, multi-constellation GNSS receiver module. [online], 2021. Available at https://www.septentrio.com/en/products/gnss-receivers/rover-base-receivers/receivers-module/mosaic.
56. Tallysman®. VSP6037L VeroStar™ Full GNSS Precision Antenna plus L-band. [online], 2021. Available at https://www.tallysman.com/product/vsp6037l-verostar-full-gnss-antenna-l-band/.
57. Uputronics™. Raspberry Pi GPS/RTC Expansion Board Datasheet, Revision 2.3. [online], 2021. Available at https://store.uputronics.com/files/Uputronics%20Raspberry%20Pi%20GPS%20RTC%20Board%20Datasheet.pdf.
58. Adafruit Industries. Ultimate GPS HAT for Raspberry Pi. [online], 2018. Available at https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ultimate-gps-hat-for-raspberry-pi.pdf?timestamp=1627027424.
59. The NTP (R&D) Project. ntpq - standard NTP query program. [online], 2018. Available at http://doc.ntp.org/current-stable/ntpq.html.
60. The GPSD project. gpsmon(1) Manual Page. [online], 2021. Available at https://gpsd.io/gpsmon.html.

Short Biography of Authors

Davide Margaria received his B.Sc. and M.Sc. degrees in Telecommunication Engineering from Politecnico di Torino in 2003 and 2007, respectively. He currently works as a senior researcher within the Connected Systems & Cybersecurity research domain of LINKS Foundation. His current research activities focus on security 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 mitigation of multipath, jamming and spoofing signals. Throughout his career he has also held a teaching role at Politecnico di Torino as lecturer under grant. He is co-author of 60+ articles on peer-reviewed scientific journals and conference proceedings.

Andrea Vesco, received the M.Sc. degree in Telecommunication Engineering and the Ph.D. in Computer and System Engineering from the Politecnico di Torino in 2003 and 2009 respectively. After one year of post-doc with the Control and Computer Engineering Department as a member of the Computer Networks Group at the Politecnico di Torino, he joined the Networking Lab at the Istituto Superiore Mario Boella (ISMB) in 2010. His research interests were on Quality-of-Service (QoS) over packet switched networks and wireless access networks. Moreover he carried out research activities on QoS over Network-on-Chip (NoC) in collaboration with the System-Level Design Group at the Columbia University of the city of New York. He had the opportunity to focus his researches on Digital Smart Cities from 2013 to 2018. Today, he is 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 IoT and critical infrastructures.