Evaluation of PTP Security Controls on gPTP

Mahdi Fotouhi*, Alessio Buscemi*, Florian Jomrich†, Christian Koebel†, Thomas Engel*
*Faculty of Science, Technology and Medicine, University of Luxembourg, †Honda R&D Europe (Germany) GmbH, *{name.surname}@uni.lu, †{Name_Surname}@de.hrdeu.com

***Abstract*—In recent years, the scientific community has been focusing on deterministic Ethernet, which has helped drive the adoption of Time-Sensitive Networking (TSN) standards. Precision Time Protocol (PTP), specified in IEEE1588 [1], is a TSN standard that enables network devices to be synchronized with a degree of precision that is noticeably higher than other Ethernet synchronization protocols [2]. Generic Precision Time Protocol (gPTP) [3], a profile of PTP, is designed to have low latency and jitter, which makes it suitable for industrial applications. However, like PTP, gPTP does not have any built-in security measures. In this work, we assess the efficacy of additional security mechanisms that were suggested for inclusion in IEEE 1588 (PTP) 2019 [1]. The analysis consists of implementing these security mechanisms on a physical gPTP-capable testbed and evaluating them on several high-risk attacks against gPTP [4].**

***Index Terms*—PTP, gPTP, Cybersecurity, TSN, Time Synchronization, Security Controls, Authentication**

I. Introduction

Time-Sensitive Networking (TSN) is a set of standards that address the time-sensitive transmission of data over deterministic Ethernet networks [5]. TSN is revolutionizing the world of industrial automation by making it more efficient, simpler to maintain, and more cost-effective. Its deterministic communication helps contribute to the ultra-reliability that is required for highly demanding industrial systems [6]. However, some of the protocols belonging to TSN lack security controls, making them, thus, exposed to a variety of attacks.

IEEE 1588 [1], known as Precision Time Protocol (PTP), is a time synchronization protocol that distributes clocks using a master-slave topology. It is composed of multiple nodes, or *clocks*, and one or more network segments (Section 6 of [1]). In PTP, a device with only one network connection is known as an Ordinary Clock (OC). This device can serve as either the source (master) or the target (slave) of a synchronization reference. A Boundary Clock (BC) has multiple network connections and it is used to synchronize one network segment to another. A Transparent Clock (TC) is neither a master nor a slave; rather, it relays PTP event messages to other clocks.

The Grand Master (GM) is the clock that serves as the root timing reference and is responsible for transmitting synchronization information to all clocks within its network. The GM can be statically configured or elected iteratively by the Best Master Clock Algorithm (BMCA). This algorithm assigns each clock a rating depending on its properties, such as its priority number, class, etc. A BC transmits precise time to the segments to which it is linked. In order to correctly manage and synchronize a PTP network, the following messages must be exchanged, i) Event messages, ($Sync$, $PDelay\_Req$, and $PDelay\_Resp$), ii) General messages ($Announce$, $Follow_Up$ $Delay\_Resp$, $PDelay\_Resp$, and $PDelay\_Resp\_Follow\_Up$).

Generic Precision Time Protocol (gPTP) (IEEE 802.1AS [3]) is the PTP profile for Audio Video Bridging (AVB) and TSN released in 2011. gPTP imposes some major restrictions to the array of possible PTP configurations. Differently from PTP, which allows switches operating at different layer of the Open Systems Interconnection (OSI) pile, gPTP requires that every switch in the network supports it at the MAC layer. Furthermore, in gPTP, there are no TCs, but only BCs. Also, in gPTP only time-aware systems can communicate gPTP information directly with one another, i.e. gPTP data cannot be transmitted over bridges that are not time-aware. In PTP, by contrast, it is possible to use non-PTP-aware bridges in a PTP domain, but doing so will delay timing convergence and introduce additional jitter that must be filtered by any PTP clock. These configurations restriction make gPTP more resistant to high delay variation when compared to PTP. As a consequence, the gPTP profile is particularly well-suited for use in industrial communication.

PTP and gPTP were designed without any security controls, making them vulnerable to various attacks, which are detailed in RFC 7384 [7]. PTP version 2.1 [8] includes an experimental security enhancement in Annex K – an optional security mechanism consisting of a Three-way handshake key distribution system. Due to the inefficiency of this mechanism's security in relation to the added overhead [9], a new optional security solution based on multiple approaches was proposed in 2019. However, it has been proven that the suggested improvement is not sufficient to prevent a number of attacks on the protocol [10]–[12].

PTP security has been examined, albeit primarily in simulation environments, which may conceal potential issues. In addition, little research has been undertaken on the security of gPTP. Therefore, it is unclear to what degree the security controls proposed for PTP are viable for usage with gPTP. To fill this gap, we implemented and tested security controls on a physical gPTP-capable testbed. The main contributions of this paper can be summarized as follows:

- We survey the advantages and disadvantages of security mechanisms for PTP proposed in related work;
- We provide a detailed description of our physical testbed with gPTP-compliant security controls;
- The security controls are evaluated by assessing the impact of high-risk attacks against our secured testbed.

II. Background

In this section, we present the current status of PTP security control measures and we discuss related work on the subject. According to PTP v2.1 [8], security control mechanisms can

28th IEEE International Symposium on Computers and Communications (ISCC 2023)

be divided into i) Integrated Security Mechanisms, ii) External Security Mechanisms, iii) Architectural Security Mechanisms, iv) Monitoring and Management Mechanisms, and v) Key Management.

A. Integrated Security Mechanisms

In this approach, an additional field, the AUTHENTICATION Type, Length, Value (TLV), is appended to the PTP messages in order to provide source authentication and message integrity and prevent replay attacks [1]. The integrated security mechanism aims to validate PTP protocol messages through two TLVs:

1) Immediate Security Processing: the AUTHENTICATION TLV is checked before processing the content of the PTP message. This processing maintains the integrity of the message and protects the whole message against unauthorized changes while intermediate nodes are able to change the mutable fields (for example, the $CorrectionField$). Every intermediate node that changes the PTP message shall recalculate and update the Integrity Check Value (ICV) in the AUTHENTICATION TLV before sending the message to the egress ports. Moreover, the ICVs are calculated with HMAC SHA256 (see page 330 of the PTP v2.1 [8]) over the whole PTP message, including header, payload and all the TLVs. Furthermore, Immediate Security Processing is not secure enough in the case of group key disclosure or an internal attack (see Section IV-A).

2) Delayed Security Processing: it is an End To End (E2E) authentication mechanism that aims at message integrity and source authentication. In this method, the key is disclosed in a different PTP message after the protected PTP message is sent. The Delayed Security Processing has been proposed in Authenticated and Unauthenticated modes, as documented in page 336 of PTP v2.1 [8].

In the Authenticated mode, the PTP messages are authenticated before processing, which means that the slaves buffer messages until they have received the corresponding key, and process them in case of a successful verification. By contrast, in the Unauthenticated mode, slaves process messages as soon as they arrive and, subsequently, the messages are buffered. The verification process takes place after the key is disclosed. In this mode, if the verification process has failed, slaves should roll back the process. Since a PTP message consists of mutable fields, such as $CorrectionField$, $ClockIdentity$, and $sequenceId$ (see page 333 of PTP v2.1 [8]), this mechanism cannot protect it as a whole. Therefore, for the calculation of the Integrity Check Value, the value of mutable fields is assumed to be zero and ignored by the receiver.

B. External Security Mechanisms

IEEE 1588 suggests using some security mechanisms, such as MACSec and/or IPSec, which were not originally included in PTP, but can be used to address some security requirements.

1) MACSec: IEEE 802.1AE Media Access Control (MAC) Security, also known as MACSec [13], provides authentication, message integrity and encryption between ports on the second layer of OSI Model [14]. Using MACSec to secure a time protocol, however, presents some obstacles. In *one-step mode* (see page 485 of [8]), for instance, the timestamp can be measured prior to decryption on the ingress port but not the egress port. Therefore, the $CorrectionField$ should be set before encryption (see page 485 of [8]). Encryption and decryption after timestamping can cause variable delays, which does not satisfy the PTP requirement for minimal delay variations (see page 485 of PTP v2.1 [8]). For MACSec to protect PTP without degrading accuracy, the latency variation must be constrained, and PTP must be used in the two-step mode that does not require on-the-fly timestamping.

2) IPSec: IPSec is a security architecture for the Internet Protocol defined in RFC 4301 [15] and RFC 4303 [16], which aims at providing security over the third layer of the OSI Model. By adding authentication and encryption, IPSec can help to prevent eavesdropping, replay attack, and message modification attack [17]. In the case of PTP, if a shared IPSec tunnel is used for both PTP messages and other network messages, the PTP messages are encrypted by IPSec. This approach can be used in networks that do not require TCs or BCs. Additionally, it can be employed in the case of a fragmented PTP network, when a PTP instance tries to synchronize to the GM through a public non-PTP aware network. For networks that rely on TCs or BCs, using IPSec may increase latency and reduce time synchronization quality [18].

There are problems in implementing IPSec on PTP aware networks. For example, it is not possible to identify incoming PTP messages for timestamping because they are encrypted and, therefore, it is not possible to distinguish them apart from other network messages. Moreover, while using hardware-based timestamping in the egress ports, it is not possible to modify the message encrypted by IPSec [11]. Thus, IEEE 1588 suggests using the *two-step* mode. It is to be noted that gPTP only works on the data link layer and the layer 2 switches are not capable of network layer protocols. Hence, IPSec cannot be used on gPTP. Two-step mode means that every $Sync$ packet should be followed by a $Follow\_Up$ packet which carries the $Sync$ packet's egress timestamp.

C. Architectural Security Mechanisms

Despite cryptographic protection, packet-based time transfer protocols such as PTP are susceptible to threats such as delay attacks. However, PTP offers architectural solutions for threat detection and mitigation (see page 490 of PTP v2.1 [8]). The primary architectural security mechanism discussed in PTP is redundancy. IEEE 1588 proposes three types of redundancy:

1) Time Source Redundancy: To increase PTP's resilience against network errors, equipment failures, and security threats, it is recommended to deploy multiple domains with multiple GMs. Multiple clocks are configured as GM instances in this method, with each clock providing accurate time for a distinct PTP domain. To receive time from multiple GMs, the slave devices, the slave devices have multiple PTP ports in slave mode, and, with three or more domains, a voting algorithm can be used to identify and counteract attacks.

2) Redundancy by complementary timing systems: it indicates that, in addition to PTP, a device receives time from other non-PTP time transfer mechanisms, such as GPS and the local

clock. In this way, a voting algorithm can identify a suspicious time source.

3) Path Redundancy: The precise time from a GM node is provided to slaves from various domains and interfaces using this method. In this instance, the slaves are provided with the exact time via multiple routes, and threats can be identified and controlled via voting algorithms.

It is to be noted that these features should not be confused with the redundancy provided by 802.1CB Frame Replication and Elimination for Reliability (FRER) [19], whose primary goal, instead, is the replication of packets for the link and node failure tolerance.

D. Other security measures

In PTP, *monitoring and management* mechanisms are considered complementary security tools (see page 492 of PTP v2.1 [8]). As an example, the network performance monitoring of PTP can be integrated into Intrusion Detection Systems (IDSs). Monitoring of jumps in the offset between the slave and the GM clock, drifts in the clock frequency and link delays are examples of parameters proposed in PTP to detect anomalies [20].

Key Management is currently not covered by PTP. It would be required to manage and distribute keys and other security parameters in the authentication and message verification process. The communication type can determine the type of the key distribution mechanism. For example, in Immediate Security Processing, group-based key distribution is used for multicast communications, while the pair-wise approach is employed for unicast communications. Also, depending on the type of security mechanism used (Immediate or Delayed), the key distribution may occur before or after the sending of PTP messages. Two main solutions for key management have been proposed in the literature, Group Domain of Interpretation (GDOI) in RFC 6407 [21] and Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in RFC 4082 [22].

E. Attacks VS Security Controls

In 2019, Shereen et al. [23] showed that when only the Immediate Security Processing is applied, an attacker can access the shared key by compromising any node in the group, thus being able to manipulate messages. In this case, a MITM attacker can make changes in the $CorrectionField$ that cause undetectable False Time in slave nodes. By contrast, changes in the payload (for example, the timestamp) can be detected by the Delayed Authentication Mechanism.

According to the authors, utilizing the Immediate Security Processing in conjunction with the Delayed Security Processing causes a sevenfold increase in the amount of delay that is experienced in TC. This is because the ICV calculation includes a portion of the message when the Immediate Mechanism is supplemented by the Delayed Mechanism. Yet the delay has no effect on the synchronization or performance of the protocols.

The IEEE 1588-2019 [8] states that the unauthenticated Delayed Security Processing is not sufficient to be used alone. As a consequence, additional security mechanisms must be added for further protection, such as mechanisms to prevent abnormal shifts in the reference time.

III. Implementation of Security Controls on gPTP

In this section, we describe the testbed we developed to evaluate security controls that we deem suitable for gPTP.

A. Identification of suitable security controls for gPTP

Identifying appropriate security measures for gPTP was the initial phase of our work.

The External Security Measures, introduced in Section II-B, include encryption and decryption, which causes variable delays, whereas PTP must produce the smallest delay variation possible. Implementing IPSec on PTP-aware networks presents some challenges, such as the inability to identify incoming PTP messages for timestamping. Furthermore, messages encrypted by IPSec cannot be modified when using hardware-based timestamping in the egress ports [11].

In the context of the redundant architectural mechanisms discussed in Section II-C, a main limitation is that redundant clocks might be less accurate than the main clocks. Furthermore, we argue that, as of today, the addition of redundancy in time synchronization seems to be not enough practicable in industrial applications, e.g. in-vehicle communication, which is one of the gPTP use cases, due to costs and complexity.

Monitoring systems such as IDSs (see Section II-D) have proven useful in identifying unusual behavior across a variety of networking protocols and circumstances. However, they are not designed for prevention, but rather for the response, and their use should supplement existing security measures. Finally, for what concerns Key Management (see Section II-D), in many of the current industrial scenarios keys are often pre-shared by injection during production. Depending on the evolution of manufacturing capabilities, this practice might change in the future, thus making the adoption of a Key Management Mechanism necessary.

We decided, based on the motivations presented in this section, that the primary security control to defend gPTP would be Integrated Mechanisms (see Section II-A, whose principal advantages are their low complexity and overhead. In this section, we describe the testbed that we set up for the implementation of the Integrated Security Mechanisms.

B. Preliminary Testbed Benchmarking

Integrated security controls have not been developed in any open-source PTP stack or daemon currently available (e.g. OpenAvnu, PTPLinux, and ptp4l). In their paper, Shereen et al. [23] is the first to conduct an implementation of security controls for PTP. Unfortunately, this implementation has not been publicly released.

Furthermore, we have examined *Domain Time II*, a commercial synchronization system [24] implemented on PTP v2.1. It includes integrated security controls and supports network auditing, monitoring, and management. Finally, the Domain Time II system operates only on the network layer (L3) and is not compatible with gPTP which works only at the data link layer (L2). For the aforementioned reasons, we have decided to employ LinuxPTP, an open-source repository for the development of PTP on Linux [25]. Given that LinuxPTP is

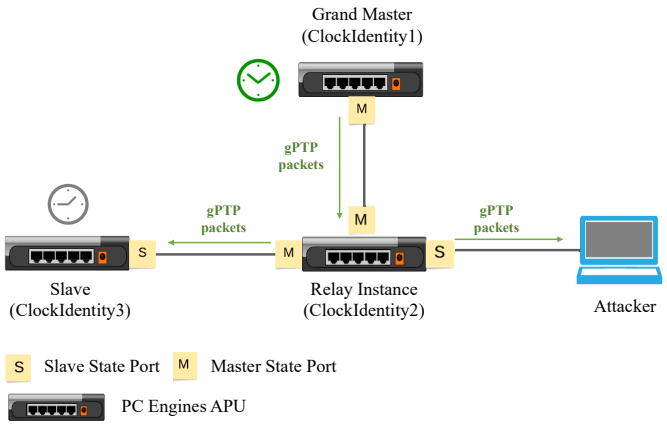

Figure 1. The figure illustrates the testbed employed in this work. In particular, it is shown the normal state of the network, i.e. gPTP is not under attack. After receiving the gPTP packets from the GM, the relay instance relays gPTP packets to the other nodes.

based on PTP, we have worked on the implementation of PTP security controls, but carefully considering gPTP characteristics to ensure compatibility with this profile.

C. Testbed

In this work, we built a physical testbed, shown in Figure 1, with the following characteristics:

- Three Accelerated Processing Units (APUs) apu2e4 [26] each with three Intel Ethernet Controller i210 and running Ubuntu 16.04. One of the APUs was configured to work as a gPTP relay instance, while the other two APUs were configured to work as the GM and the slave endpoints respectively. Each APU mounts LinuxPTP (version 3.1-00116-g24220e8).
- One laptop running LinuxPTP as the attacker's device, connected via RJ45 standard Ethernet connectors.

We make use of the real-time report that is produced by the LinuxPTP daemon while it is operating, in order to monitor clock offsets and delays. In addition, we monitor the states of the ports on the switches, the joining status to the gPTP domain (whether the port is enabled or disabled), the propagation delay, and the number of gPTP packets that are dropped by the switch. This is done through the control panels of the switches. Finally, Tcpdump is utilized at each endpoint in order to sniff packets that are either transmitted to or received by a device.

D. Security Controls for gPTP

Based on the results of our preliminary investigation, due to the peculiarities of the gPTP profile, PTP Delayed Security Processing (E2E authentication) cannot be fully applied to gPTP. As an example, in addition to the $CorrectionField$, gPTP relay instances do not preserve many gPTP message fields like $ClockIdentity$, $SequenceNumber$, and TLVs (path trace TLV or $Follow\_Up$ TLV). Additionally, the PTP Immediate Security Control is not effective in the cases of a group key disclosure or an internal attack. Figure 2 shows the structure of the $Announce$ message. The figure highlights that the PTP mutable fields (represented in red) cannot be covered by the E2E Authentication Mechanism. Therefore, while an *external* attacker is unable to modify these mutable fields, an *internal* attacker with access to the group authentication key can manipulate or forge them undetected (see Section IV-A).

We observed that LinuxPTP does not accept TLVs on all messages. For this reason, we focus our emphasis to establish security controls on the realisation of the AUTHENTICATION TLV on $Announce$ and $Follow\_Up$ messages only. This choice is motivated by the essential role that these messages have for the communication in PTP (i.e. maintaining the synchronization between GM and slaves). We leave the investigation of security control for other types of PTP messages to future research.

The structure of the AUTHENTICATION TLV was defined based on the description contained in the PTP standard. The Group AUTHENTICATION ICV is calculated over the whole gPTP message by a pre-shared group key. For gPTP compatibility, we applied the E2E authentication only on the immutable fields which are changed by the intermediate clocks. In this study, we do not consider the key disclosure mechanism proposed for the Delayed Security Processing in the PTP standard. It is to be noted that the focus of this work is limited to the authentication itself, and not the key exchange mechanism. In this context, we assume that the key disclosure mechanism is equivalent to a pre-shared key between two identities.

IV. Security Controls Evaluation

In this section, we present the attacker model and we evaluate the impact of the attacks conducted against the secured gPTP testbed presented in Section III.

A. Attacker Model

The RFC 7384 [27] defines two types of attackers, internal or external. An internal attacker is capable of accessing shared security keys, while the external attacker does not have access to them. Furthermore, internal and external attackers can occupy either Man in The Middle (MITM) or internal positions. We summarize the different combinations of attackers and positions in Table I.

Regarding the attack scenarios, we assume that an internal attacker has access to the group authentication key, but not to the E2E keys of other nodes. The attacker types and their abilities are illustrated in Figure 3. In the following subsections, we evaluate the impact of spoofing, rogue master and message manipulation attacks individually.

B. Spoofing Attack

On the testbed, we have performed two types of spoofing attacks which we describe hereafter.

1) Announce Spoofing Attack: in this attack, the attacker spoofs the $Announce$ message sent by the legitimate GM. The spoofing attack on gPTP with active integrated security controls is unsuccessful if the attacker spoofs immutable fields, such as the GM Identity or the priority, as they are protected by the E2E authentication. Moreover, even though spoofing some mutable fields, such as the $sequence\_Id$ and $CorrectionField$, is not

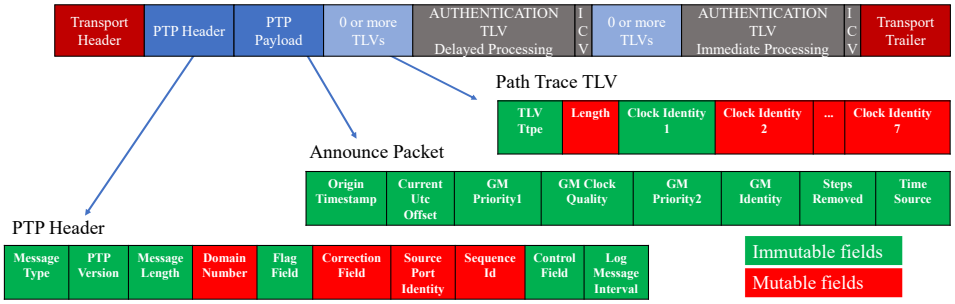

Figure 2. Mutable and immutable fields in a gPTP *Announce* message. The mutable fields are not protected by the E2E authentication mechanism.

Table I
ATTACKER MODEL

	Internal	**External**
MITM attacker	The attacker controls of at least an intermediate node in the network or has physical access to the communication channel between two nodes with knowledge of the security keys. In this position, the attacker is capable to intercept, drop, and manipulate protocol packets.	The attacker has all capabilities of an internal attacker but cannot access any security keys.
Injector	The attacker is part of the same network, but not in the middle of a communication channel. So, the injector cannot intercept, drop, and manipulate protocol packets, but is capable of injecting new packets into the network and eavesdropping on protocol packets sent by other nodes.	The attacker has all capabilities of an internal injector, but does not have access to any security keys.

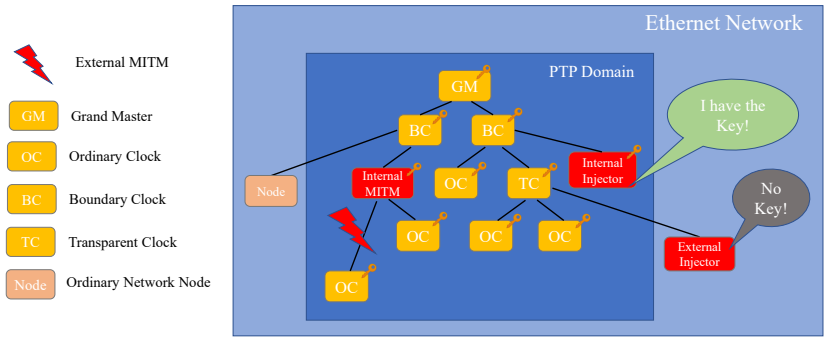

Figure 3. The figure illustrates the attacker model. In particular, it is highlighted that the Internal Injector and Internal MITM are part of the PTP Domain and, therefore, have access to the shared security keys. By contrast, the External Injector does not have access to the key.

detectable by the Integrated Security Mechanism, it does not help the attacker to produce considerable impact. Nevertheless, we observed that an internal attacker can perform a successful Denial of Service (DoS) attack if he changes the $clockIdentity$ field to an arbitrary value other than the slave's parent port's $clockIdentity$. Let us consider a scenario in which the frequency of spoofed *Announce* messages is higher than the frequency of the *Announce* message sent by the legitimate GM, as shown in Figure 4. The slaves GM reject all *Sync* & $Follow\_Up$ messages coming from the GM, given that their $clockIdentity$ is different from what previously received in the spoofed *Announce* messages. As a consequence, the slaves fail to synchronize to the GM. In another scenario related to the *Announce* spoofing attack, as the path trace TLV is not protected by the E2E AUTHENTICATION TLV, an attacker is able to replay the *Announce* messages generated by the GM while path trace information is spoofed. As demonstrated in [4], this attack causes desynchronization and DoS.

2) Sync Spoofing Attack: Spoofing *Sync* and $Follow\_Up$ messages are prohibited for an injector attacker due to port state limits. However, [4] demonstrates that the injector attacker can

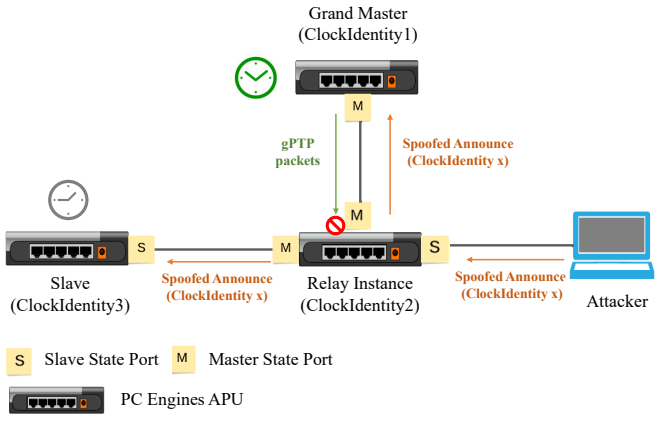

Figure 4. The figure shows an *Announce* spoofing attack causing DoS. When the attacker sends the spoofed *Announce* packet containing manipulated $ClockIdentity$, the relay instance changes the port states and blocks *Sync* & $Follow\_Up$ messages coming from the GM that causes DoS in the slaves.

bypass this limit if the gPTP messages' EtherType and MAC Address are checked incorrectly. During our examination, we observed that modifying immutable fields like the timestamp

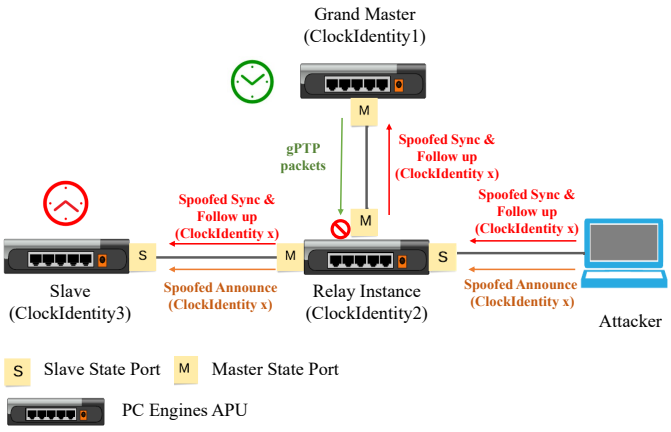

Figure 5. False Time caused by a Spoofing attack. The attacker has already injected a spoofed *Announce* packet containing a manipulated *ClockIdentity*. As a consequence, the relay instance has changed the port states and blocked *Sync* and *Follow_Up* messages from the GM. At this point, the attacker sends old *Sync* and *Follow_Up* messages with a spoofed *ClockIdentity* that causes slaves' clocks to display the incorrect time.

field does not help an internal attacker desynchronize the slaves' clock as the E2E authentication mechanism prevents the spoofed *Sync* and *Follow_Up* messages from being processed.

Moreover, our analysis shows that exploiting mutable fields such as *SequencId*, *clockIdentity*, and *CorrectionField* in *Sync* and/or *Follow_Up* messages does not enable an internal injector to produce False Time on slave clocks. Nonetheless, we demonstrate that an internal attacker is able to produce False Time in slaves by altering only *clockIdentity* fields in old *Announce*, *Sync*, and *Follow_up* messages from the GM, and sending them at a higher frequency. In this scenario, the attacker first sends spoofed *Announce* messages with arbitrary *clockIdentity*, like *new_clockIdentity* (as described in IV-B1), which causes the port state change in the intermediate clock. Consequently, all legitimate *Sync* and *Follow_up* from the real GM are rejected by the relay node. Then, the attacker causes False Time by replaying previously sent *Sync* and *Follow_up* messages with the *new_ClockIdentity* (Figure 5). In this scenario, without broadcasting gPTP messages, the attacker can successfully achieve False Time by enforcing the slaves' clock back to a time in the past and desynchronizing them from the GM.

C. Rogue Master Attack

The attacker has acquired control of a network node and is awarded GM status via the BMCA by sending *Announce* packets with a high GM priority. Once the GM position is granted, the adversary can conduct False Time, Desynchronization, and DoS attacks.

The integrated security mechanism proposed by PTP does not help to prevent rogue master attacks. It does not act on the BMCA or any member of the PTP domain who has its own E2E key and the group key to send and receive PTP messages. Therefore, an internal attacker in an injector position has the ability to generate *Announce* messages with a high priority and include AUTHENTICATION TLVs with its own security keys in order to achieve the GM position and alter the slaves' clocks. This attack has the same effect on our secured testbed as if no security measures had been deployed.

D. Message Manipulation Attack

From an internal MITM position, the attacker is able to modify the received *Announce* and *Sync/Follow_Up* messages' *clockIdentity*. In this scenario, the attacker does not need to bypass or change the port states as described in IV-B2, because he controls at least one PTP port in a master state and, therefore, can send *Sync/Follow_Up* messages without port state limits. In this context, the attacker sends old *Sync/Follow_up* messages while preventing new messages from passing. The impact of this scenario, even with the presence of E2E Authentication, is False Time and Desynchronization for all the slaves.

V. Discussion

Table II provides a summary of the impact that the security controls have in protecting against the attacks evaluated against our testbed. As highlighted in the table, the proposed security controls are effective against external attackers but insufficient to protect gPTP from internal adversaries.

We argue that internal attackers could be addressed by using a Peer-to-Peer (P2P) Authentication Mechanism. In a P2P Authentication Mechanism, each pair of nodes that are directly connected have a shared key that covers the whole gPTP message. The advantage of this mechanism is that it provides both the integrity check and the parent authentication, and can be extended to the whole domain. Additionally, mutable fields are completely protected, thus shielding the gPTP domain from unauthorized message manipulation and message spoofing attacks. It is to be noted, however, that this approach presents a major drawback concerning the overhead added by the key management. Differently from the overhead brought by the Group Based Mechanism, the overhead produced by the Delayed Authentication Mechanism's key management proposed in PTP is non-negligible.

To circumvent the competition of the packets coming from the GM, the attacker can send spoofed messages at shorter intervals than those defined in the gPTP configuration, and the PTP instances will process these packets as frequently as they arrive. In this regard, we recommend limiting incoming gPTP messages to predetermined intervals. For instance, if the *Announce* interval is defined as 1s in the gPTP configuration, there should be a control to ensure that no two *Announce* messages are processed within that interval. These restrictions would hinder the adversary's attempts to outperform the authorized GM.

VI. Conclusion

This work aimed at assessing the efficacy of security controls for gPTP against a variety of attacking scenarios, presented in Section IV-A. For the evaluation, we integrated security controls proposed in the PTP standard on a physical testbed integrating LinuxPTP, in compliance with the gPTP profile's requirements and constraints.

Table II
EVALUATION OF THE SECURITY CONTROLS EFFICACY AGAINST THE TESTED ATTACKS

Security controls	External Spoofing attack (Inj+MITM)	Internal Injector Spoofing attack	Rogue master attack	Internal Message Manipulation Attack / Internal MITM Spoofing
PTP Group Authentication	Prevents	Cannot prevent	Cannot prevent internal attacks	Cannot prevent
PTP E2E authentication	Cannot prevent the attack on gPTP if mutable fields are spoofed	Cannot prevent spoofing mutable fields	Cannot prevent internal attacks	Cannot be prevented on gPTP if mutable fields are manipulated/spoofed
E2E + Group authentication	Prevents	Cannot prevent spoofing mutable fields	Cannot prevent internal attacks	Cannot be prevented on gPTP if mutable fields are manipulated/spoofed

According to the findings of our research, these security mechanisms are effective in protecting against external spoofing attacks but are unable to defend against an attacker with an internal position. An internal injector attacker can take advantage of changeable fields in gPTP messages to generate False time, cause desynchronization, and deny service to slave clocks, as well as misuse the mutable $clockIdentity$ field in the gPTP messages to conduct a successful spoofing attack.

The reason is to be found in the lack of protection of the mutable fields contained within gPTP messages, as the E2E authentication does not fully cover all the fields contained in them. Furthermore, the Group Authentication Mechanism cannot protect messages from unauthorized modification, thus allowing them to be spoofed by an internal attacker either in the injector or MITM position.

To address the drawbacks of the presented security controls, we discussed the possibility of using a P2P Authentication Mechanism. The implementation and evaluation of the effectiveness of this security control are left to future work. Furthermore, in future work we plan to extend the architecture and capabilities of our testbed to perform a thorough analysis of additional realistic adversarial scenarios.

ACKNOWLEDGMENT

This work was supported by the BRIDGES grant, funded by the Luxembourg National Research Fund (FNR), and by Honda R&D Europe (Germany) GmbH.

REFERENCES

[1] IEEE Std 1588™-2019, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE SA, Standard, 2019.

[2] L. Deng, G. Xie, H. Liu, Y. Han, R. Li, and K. Li, "A survey of real-time ethernet modeling and design methodologies: From AVB to TSN," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–36, 2022.

[3] "IEEE Draft Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," *IEEE Draft Std P802.1AS/D2.0 Feb 2008*, 2008.

[4] M. Fotouhi, A. Buscemi, A. Boualouache, F. Jomrich, C. Koebel, and T. Engel, "Assessing the Impact of Attacks on an Automotive Ethernet Time Synchronization Testbed," *2023 IEEE Vehicular Networking Conference (VNC)*, 2023.

[5] N. Finn, "Introduction to time-sensitive networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 22–28, 2018.

[6] L. L. Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.

[7] M. Mizrahi, "RFC 7384: Security requirements of time protocols in packet switched networks," *Tools. ietf. org (online) https://tools. ietf. org/html/rfc7384 (accessed 26 Sep 2020)*, 2014.

[8] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)*, pp. 1–499, 2020.

[9] C. Önal and H. Kirrmann, "Security improvements for IEEE 1588 Annex K: Implementation and comparison of authentication codes," in *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*, IEEE, 2012, pp. 1–6.

[10] E. Itkin and A. Wool, "A security analysis and revised security extension for the precision time protocol," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 1, pp. 22–34, 2020.

[11] W. Alghamdi and M. Schukat, "Advanced methodologies to deter internal attacks in PTP time synchronization networks," in *2017 28th Irish Signals and Systems Conference (ISSC)*, IEEE, 2017, pp. 1–6.

[12] N. Moreira, J. Lázaro, J. Jimenez, M. Idirin, and A. Astarloa, "Security mechanisms to protect IEEE 1588 synchronization: State of the art and trends," in *2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, IEEE, 2015, pp. 115–120.

[13] "IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security," *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pp. 1–239, 2018.

[14] J. Day and H. Zimmermann, "The OSI reference model," *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1334–1340, 1983.

[15] S. Kent and R. Atkinson, *Security architecture for the internet protocol*, 1998.

[16] S. Kent and R. Atkinson, "IP encapsulating security payload (ESP)," 1998.

[17] 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, 2021.

[18] D. T. Chen, "Secure 1588 in HeNB/Femtocell application," *Time & Sync Telecoms, ITSF*, 2013.

[19] I. S. Association, "IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability," *C/LM - LAN/MAN Standards Committee*, 2017.

[20] A. Buscemi, M. Ponaka, M. Fotouhi, F. Jomrich, C. Koebel, and T. Engel, "An Intrusion Detection System Against Rogue Master Attacks on gPTP," *2023 IEEE Vehicular Technology Conference (VTC)*, 2023.

[21] B. Weis, S. Rowles, and T. Hardjono, "The group domain of interpretation," *Internet Request for Comments*, vol. 6407, 2011.

[22] A. Perrig, D. Song, R. Canetti, J. Tygar, and B. Briscoe, "Timed efficient stream loss-tolerant authentication (TESLA): Multicast source authentication transform introduction," *Request For Comments*, vol. 4082, 2005.

[23] E. Shereen, F. Bitard, G. Dán, T. Sel, and S. Fries, "Next steps in security for time synchronization: Experiences from implementing IEEE 1588 v2. 1," in *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, IEEE, 2019, pp. 1–6.

[24] greyware. "https://www.greyware.com/software/domaintime/." (2022).

[25] linuxptp, *ptp4l*. [Online]. Available: http://linuxptp.sourceforge.net/.

[26] P. Engine, *apu2e4*, May 2016. [Online]. Available: https://pcengines.ch/apu2e4.htm.

[27] T. Mizrahi, "Security requirements of time protocols in packet switched networks," in *RFC 7384*, 2014.