Precision Time Protocol - Security Requirements

Tizian Leonhardt, Filip Rezabek[*], Kilian Holzinger[*]
[*]*Chair of Network Architectures and Services, Department of Informatics*
Technical University of Munich, Germany
Email: tizian.leonhardt@tum.de, rezabek@net.in.tum.de, holzinger@net.in.tum.de

***Abstract*—The ever-growing amount of timing-sensitive applications necessitates clock synchronization that can offer guarantees of high precision. The Precision Time Protocol offers sub-microsecond accuracies for clock-based networks. However, there is sufficient evidence that attacks are a substantial threat that can have devastating consequences. In this paper, we examine security requirements in the context of the Precision Time Protocol and evaluate how they may be met by different security solutions, as well as one of its open-source implementations, `linuxptp`.**

***Index Terms*—time protocols, network security, ptp**

1. Introduction

The increasing need for clock synchronization with high precision requirements demands protocols that can achieve accuracies in the micro and nanosecond ranges. The Precision Time Protocol (PTP), standardized in the `IEEE1588` standard, can cater to these requirements. It largely surpasses the Network Time Protocol (NTP); compared to NTP delivering accuracies in the millisecond range, PTP allows for sub-microsecond accuracies [1]. This is achieved with timestamps at the hardware level, effectively bypassing any noise that would be introduced by the network stack [2].

We now want to motivate why the topic of security is worth discussing in the context of timing protocols. An obvious result of an attack is the falsification of one or more clocks in the network. The implications of this seemingly harmless effect are not to be underestimated. Smart grids, as an example, rely on accurate timestamps and often have the obligation to deliver accuracies in the microsecond range [3]. This enables them to effectively deliver electricity, a crucial resource. Attacks on power delivery can have devastating consequences [4]. Systems that rely on high accuracies are also more sensitive to attacks, as deviations have a higher influence, making PTP an attractive goal for attackers. This paper aims to analyze the requirements of a secure PTP environment, as well as to evaluate different security solutions concerning these demands.

In Section 2, we first lay the technical foundation needed for understanding PTP, as well as its different versions. This also entails a discussion of the aforementioned security requirements. Section 3 contains the analysis of a handful of security solutions in the context of different attack scenarios, the results of which are summarized in a table. In Section 4, we compare the previous results with `linuxptp`, an open-source implementation of PTP. Section 5 concludes the paper and gives an outlook on future work.

2. Background

In this section, we discuss the technical intricacies surrounding PTP, as well as its different versions. We also review the security requirements defined in RFC7384 [5].

2.1. Precision Time Protocol

The following findings are, unless otherwise noted, based on [1]. As already stated, PTP allows the synchronization of multiple clocks in a network with high precision. The protocol is made up of a multitude of clocks serving different purposes, laid out in a master-slave hierarchy. Figure 1 seeks to give an overview of this. Ordinary clocks (OC) have one external port and act

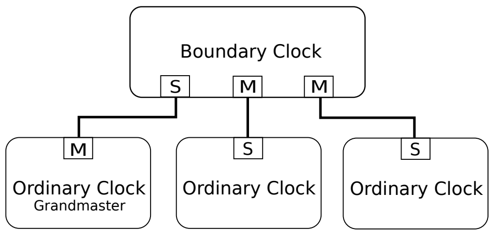

Figure 1: A master-slave clock diagram [1, 6.6.2.4]. 'M' marks a port as master, 'S' as slave. The grandmaster clock is highlighted.

as either master or slave. A boundary clock (BC) on the other hand features n ports, where $n > 1$. It is responsible for synchronizing the network segment it governs and on one port listens as a slave for the synchronization of its own clock. The BC propagates this clock to the remaining $n - 1$ ports associated with clocks in the segment. While more variants of clocks exist, we only presented the ones we deem necessary for a general understanding of a PTP network.

In the case of Figure 1, the OC marked grandmaster presents a special case. This clock is responsible for propagating the time reference and therefore establishes the idea of time in the system. This reference can be fetched from a reliable external source, such as a GPS signal. The grandmaster clock is dynamically chosen by the Best Master Clock (BMC) algorithm, which picks the best candidate according to various criteria, such as the quality of the time source; the candidates have to announce their parameters in order to register for this election.

The IEEE1588 standard also defines a way to deal with cyclic paths in mesh topologies. To avoid synchronizing a BC from multiple sources, superfluous paths are removed by setting the corresponding slave port to passive; this prevents any timing information from being exchanged. Figure 2 illustrates this with an example where one path is pruned, resulting in a tree structure. Note that only the ports necessary for this example have a connection. Besides electing the most suitable master clock to be the grandmaster, the BMC algorithm is also responsible for selecting the path to be excluded.

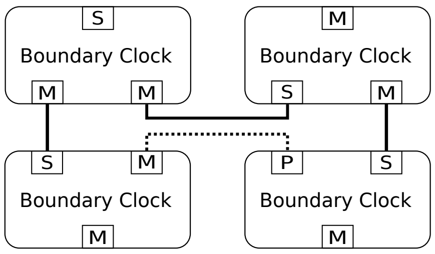

Figure 2: An example of mesh topology pruning [1, 6.6.2.5]. The pruned path is dashed. 'P' marks a port as passive.

The second PTP component we consider is the mechanism responsible for synchronizing the clocks. Propagating the grandmaster clock value itself is trivial, but the delay between the master and its slaves also has to be accounted for. This is determined with a sequence of protocol messages that compute the delay. When a master initiates the synchronization, the slave denotes the time when the message arrived, which is followed up by the master transmitting the time of his initial request. This is subsequently done in the direction of the slave to his master as well. With this information, the correct offset is computed. It should be noted that this process relies on the central assumption that the delay between master and slave is equal in both directions, i.e., the paths are symmetric [1, 6.2].

2.2. PTP Versions

Currently, three versions of the IEEE1588 standard exist. The 2002 version is not of interest for this paper as it is outdated and incompatible with the newest revision [6]. In contrast, the 2008 revision remains largely compatible with the newest standard [6]. During the last twelve years, many issues with this version have been identified [2], [7]. This raises the need for an improved standard, which is now released as revision 2019 and aims to fix many of the aforementioned issues. Besides that, there is also the IEEE standard 802.1AS, an adoption of the IEEE1588 standard to better accommodate to time-sensitive audio and video traffic [8]. The remaining sections of the paper revolve around the two latest IEEE1588 versions.

2.3. Security Requirements

Based on the previous insights, it is discernible that we need to protect against attacks to warrant the security of time-critical systems. This is a relevant topic to PTP, not just as a result of its high accuracy demands, but also because security was not a main concern during the design of the first two revisions [9]. For the 2008 version, an experimental annex ('Annex K') to the standard exists, providing "group source authentication, message integrity, and replay attack protection for PTP messages." [1, K1] On top of its experimental nature, multiple sources state the obsoleteness of this annex [2], [10], which is why we pay little attention to it going forward.

In order to better understand the demands of a secure PTP environment, RFC7384 [5] offers a guideline by listing security requirements in various contexts. Our evaluations going forward are largely based on this RFC. We focus on the so-called MUST-types (see also [11]), i.e., requirements that *have* to be implemented to create a secure PTP environment. Unless otherwise stated, all requirements in the following sections are of this type. The relevant requirements for later parts of the paper include [5]:

- Authentication and Authorization
- Integrity protection
- Spoofing prevention
- Replay protection
- Protection against delay and interception
- Availability

'Authentication and Authorization' is concerned with uniquely identifying clocks in the network and ensuring that their respective behavior does not violate permission boundaries. The 'Integrity protection' requirement necessitates techniques to verify that messages have not been corrupted or tampered with. 'Availability' describes the protection against Denial of Service (DoS) attacks.

3. Threat Mitigation

We now highlight a selection of attacks and their respective security solutions. We also evaluate them in regard to the aforementioned requirements. In the context of the 2008 revision of PTP, we focus on security solutions that are novel to the standard. For the 2019 version, we focus on the new security features that are integrated into the standard; this features some general security considerations instead of solely focusing on a attack scenario. This section concludes with a table that presents the results in a compact form.

3.1. IEEE1588-2008

We begin the analysis with the 2008 revision. Each attack addressed is placed in a new subsection. Even though this revision is already superseded, it is still of interest for this paper due to the newness of the IEEE1588-2019 standard at the time of writing.

3.1.1. Delay Attacks. Reference [3] revolves around exploiting the assumption of symmetric paths discussed in Section 2.1. The threat model is based on an attacker with access to the internal network infrastructure. The attack is executed by delaying the messages used for computing the delay between two PTP nodes in one direction, effectively creating an asymmetry that "introduces an error in the computed value of the clock offset." [1, 6.6.3] This ultimately leads to a skewed clock, violating the requirement of 'Protection against delay and interception' [5]. Figure 3 seeks to explain this asymmetry caused by a rogue node.

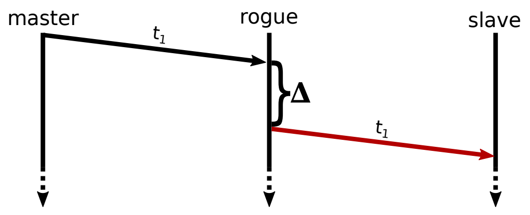

Figure 3: A rogue node intercepts and delays a synchronization message by Δ. Note that this only happens unidirectional. In [3], Δ is chosen randomly.

In order to combat this vulnerability, [3] proposes a solution based on detection and mitigation. Detecting a delay attack is made possible by installing a second, redundant clock that is retrieving its timing information from the same source as the grandmaster (e.g. a GPS signal). This node also responds to delay computation requests and calculates the new clock value; if the difference of this value is not equal to the external time reference, an attack is likely in progress. To reduce the impact of the attack, a cumulative average based on previous offsets for each node is used for calculating the clock value. The fact that this does not completely annul the effects of the delayed messages is discussed by [3], with the conclusion that this method leaves enough time for authorities to respond to the attack, as the rogue node has to be inside the network. Whether or not this is a realistic assumption is not further evaluated. With enough time, an attacker can still skew the clocks, leaving the system open to attacks if the response is not timely enough.

3.1.2. Denial of Service. Even though RFC7384 defines the protection against DoS attacks as a SHOULD-requirement ('Availability') [5], the low amount of effort needed for the DoS-attack demonstrated by [9] is sufficient evidence for treating it as an important requirement that should not be overlooked. The technique demonstrated relies on forging spurious synchronization packets that are sent to slaves at a rate of around 292 packets per second. The forged packets contain correct identification details (e.g. the clock ID) for the corresponding master node, which can be obtained by sniffing the traffic; the semantics of the synchronization itself are non-existent, as they are not needed for the attack to succeed. After gathering this, the packets can be sent without any knowledge about the slaves through the fixated multicast address. The setup for the attack is therefore comparatively simple, and indeed, [9] reports delays of multiple hours in the test environment by overburdening the nodes with the forged traffic. Even though further tests in real-world PTP networks would be necessary to assess the actual impact of the attack, it still is a significant result.

Just as discussed in Section 3.1.1, [9] proposes solutions based on mitigation, as well as detection. For the former, multiple approaches are suggested. We only discuss the introduction of a digital entity, as the other methods are not further evaluated. This digital entity is introduced for master nodes, opening up the possibility of identifying themselves through cryptographic means. Utilizing this, nodes are able to filter packets based on whether or not they "originate from masters with a valid identity." [9] The desired effect is furthermore confirmed, substantiating complete protection from the demonstrated attack. We note that the specific implementation of the digital entity is out of the scope of this paper, but it is not guaranteed that other approaches would yield the same efficacy.

3.1.3. Best Master Clock Spoofing. The Best Master Clock algorithm [6], as stated in Section 2, chooses the best clock out of a set of potential masters to act as the grandmaster. This procedure, at its core, compares software-defined quality parameters that clocks announce and acts accordingly. The parameters include, but are not limited to [6, 6.6.2.3]:

- `priority1`
- `clockClass`
- `clockAccuracy`

The value `priority1` is an integer chosen by the administrator to allow for own priority suggestions, whereas `clockClass` categorizes clocks into further subcategories. Especially interesting is `clockAccuracy`, which provides an upper bound for the accuracy offered by the individual clocks. Accuracy bounds range from more than ten seconds down to one picosecond [6], several orders of magnitude smaller than the performance advertised by the standards surrounding PTP. It is apparent that there is a significant potential for abuse by spoofing values that no clock in a real-world scenario would offer. The need for protection against this kind of attack is described by the 'Spoofing Prevention' requirement [5].

This attack is successfully demonstrated in [2]. Two types of attackers are considered: an external attacker that can only see the public multicast traffic, as well as an internal attacker, which is also a node of the PTP network. Both approaches make use of setting a selection of the previously discussed quality parameters to the best possible values. Announcing these parameters guarantees a win in the election and therefore control over the time propagation. Reference [2] suggests the use of symmetric cryptography in order to mitigate the attack from an external standpoint; this is also the elected method in Annex K, which is applicable here [2], despite its flaws. In contrast, the internal attacker, as part of the PTP network itself, would know the secrets of a symmetric encryption. To counter this, the employment of asymmetric cryptography is suggested, which is confirmed as an effective measure. Reference [9] also discusses this technique, but extends it by closely mimicking the behavior of other master clocks in the network; the details are gathered through sniffing.

3.2. IEEE1588-2019 - Annex P

As a replacement for the obsolete Annex K, the 2019 revision of PTP includes a new security model on which this section is based, defined in Annex P [6]. This is based on four prongs [6], each serving a different purpose in terms of security. The standard even acknowledges RCF7384, stating that the approach presented is tied to the requirements mentioned there. The prongs are made up of the following concepts:

A Integrated Security Mechanism
B External Transport Security Mechanisms
C Architecture Mechanisms
D Monitoring and Management Mechanisms

While all of these prongs play an important role, we only focus on prongs A and D in this paper, as they are the closest to the protocol itself. Prong B is concerned with more general networking techniques that could increase PTP security (for example MACsec), whereas prong C discusses topology enhancements, such as redundancies for master clocks.

The integrated security mechanism of prong A employs symmetric cryptography by adding 'type-length-value' (TLV) attributes to the protocol messages. They allow the direct extension of messages with attributes of arbitrary length. For `IEEE1588-2019`, the so-called 'Authentication TLV' provides "source authentication, message integrity, and replay attack protection [...]." [6, 16.14] To do so, the TLV carries all the necessary data to enable the secure processing of messages, such as the 'Integrity Check Value' (ICV) that is used to verify the integrity of a message. The concept of authentication by adding a TLV is also present in Annex K of the `2008` revision with the same goals in mind. Prong A therefore already addresses 7 out of the 12 MUST-requirements found in RFC7384 in a cryptographically sound way. The feasibility of this approach is asserted by [7], confirming that the accuracy of PTP is not negatively impacted. Although prong A addresses many of the requirements, PTP is still possibly open to delay attacks [7] (besides potential others). This is one part of the issues that prong D should address, though the general responsibilities are much broader and can be tailored to fit the needs of the underlying system. One possible way to combat delay attacks has already been presented in Section 3.1.1, which could be implemented for the newest revision as well. A similar approach that is based on prong D is discussed in [12].

3.3. Results

We conclude this section with Table 1, allowing for a convenient point of reference. It contains an overview of the requirements addressed by the references that were cited in Section 3. Note that this table might include additional details about contributions that were not discussed earlier. An x means addressed, a dash means not addressed. References [1] and [6] refer to Annex K and P, respectively.

4. Case Study: linuxptp

Having reviewed a multitude of security solutions, we now shift our perspective towards the available security features of `linuxptp` [13]. This is based on a comparison of the insights already garnered, as well as security features not previously mentioned. We first discuss the supported security TLV types, defined in `tlv.h`. Although the `AUTHENTICATION` TLV introduced in Annex P [6] is present, it is simply ignored during the processing of protocol messages [13, tlv.c]. The same applies to the

TABLE 1: Comprehensive overview of requirements [5] addressed by various contributions

	Addressed Requirements					
	[3]	[9]	[2]	[12]	[1]	[6]
Authentication and Authorization	-	x	x	-	x	x
Integrity protection	-	-	x	-	x	x
Spoofing prevention	-	x	x	-	x	x
Replay protection	-	-	x	x	x	x
Protection against delay and interception	x	-	x	x	-	x
Availability	-	x	-	x	-	x

authentication TLVs that are used in Annex K [1], leaving authentication through those means impossible without additions to the code. Further research reveals that no other options for authentication currently exist.

A comparatively simple way of checking for attempts at skewing clocks is to compare the value used for offsetting the clock to a maximum and minimum value; either being exceeded could hint at a possible attack. `linuxptp` reacts to unexpected jumps by issuing a warning and returning from the corresponding function with an error value [13, clockcheck.c]. While this does catch obvious attacks and is mentioned as a mitigation mechanism for prong D in Annex P [6], continuously introducing delays, as demonstrated earlier in Section 3.1.1, would still go unnoticed if Δ is chosen within an appropriate range.

Another attractive attack vector are master clocks. They play the central role of synchronizing their slaves. It is therefore unwanted that rogue masters can influence clocks. This is captured by the 'Spoofing Prevention' requirement in RFC7384, which mentions authentication as a possible solution [5]. Even though no authentication mechanism currently exists in `linuxptp`, there still is a check in place to mitigate this attack. Whenever a slave processes synchronization messages (for example in `process_delay_resp` [13, port.c]), the identity of the source port is checked; should the sender of the synchronization messages not align with the currently associated master clock, the message is discarded [13, port.c]. However, as this is based on values that could be obtained by sniffing the traffic (see also [13, ddt.h]), an attacker could simply determine the correct identification for each slave node. The feasibility of sniffing traffic for this type of information is illustrated in [9].

In summary, there is a lot of work that could be done regarding security features in `linuxptp`, especially in light of the integration of the new security features found in Annex P [6]; only the integration of the authentication TLV and the surrounding techniques would address a great number of critical requirements.

5. Conclusion

We have shown that PTP is an interesting target for potential attackers that could have far-reaching consequences. Only slight deviations could influence the accuracy needed for systems that are reliant on it. We then analyzed security solutions with respect to the requirements defined in RFC7384. The results, which are also showcased in Table 1, are promising; the solutions presented are theoretically able to mitigate many critical attacks. The case study on `linuxptp` illustrated the need for better security options, such as mechanisms for authentication. This paper could serve as a solid foundation for security considerations regarding the latest two PTP versions. It also paves the way for further evaluations regarding the state of security solutions. Future work could analyze new experiences with annex P (`IEEE1588-2019`), as the standard was comparatively novel at the time of writing. There are also many opportunities concerning the design of improved security features for `linuxptp`.

References

[1] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., Jul. 2008.

[2] 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, Jan. 2020.

[3] B. Moussa, M. Debbabi, and C. Assi, "A detection and mitigation model for PTP delay attack in a smart grid substation," in *2015 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, Nov. 2015.

[4] N. Kshetri and J. Voas, "Hacking power grids: A current problem," *Computer*, vol. 50, no. 12, pp. 91–95, Dec. 2017.

[5] T. Mizrahi, "Security requirements of time protocols in packet switched networks," Internet Requests for Comments, RFC Editor, RFC 7384, Oct. 2014, last accessed on 2021/01/08. [Online]. Available: https://tools.ietf.org/html/rfc7384

[6] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., Jun. 2020.

[7] E. Shereen, F. Bitard, G. Dan, 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, Sep. 2019.

[8] M. D. J. Teener and G. M. Garner, "Overview and timing performance of IEEE 802.1as," in *2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, sep 2008.

[9] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. A. Wojciak, and S. Guendert, "Impact of cyberattacks on precision time protocol," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 5, pp. 2172–2181, May 2020.

[10] D. Maftei, R. Bartos, B. Noseworthy, and T. Carlin, "Implementing proposed IEEE 1588 integrated security mechanism," in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, Sep. 2018.

[11] S. Bradner, "Key words for use in rfcs to indicate requirement levels," Internet Requests for Comments, RFC Editor, BCP 14, Mar. 1997, last accessed on 2021/01/08. [Online]. Available: https://tools.ietf.org/html/rfc2119

[12] W. Alghamd and M. Schukat, "A detection model against precision time protocol attacks," in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, Mar. 2020.

[13] R. Cochran, "linuxptp," version 3.1; last accessed on 2021/01/07. [Online]. Available: https://sourceforge.net/projects/linuxptp/