Towards an Increased Detection Sensitivity of Time-Delay Attacks on Precision Time Protocol

LEA SCHÖNBERGER¹ (Graduate Student Member, IEEE), MOHAMMAD HAMAD², JAVIER VELASQUEZ GOMEZ³, SEBASTIAN STEINHORST² (Senior Member, IEEE), and SELMA SAID¹ (Member, IEEE)

¹TU Dortmund University, 44221 Dortmund, Germany (e-mail: firstname.lastname@tu-dortmund.de)
²Technical University of Munich, 80333 Munich, Germany (e-mail: firstname.lastname@tum.de)
³Hamburg University of Technology, 21073 Hamburg, Germany (e-mail: javier.velasquez@tuhh.de)

Corresponding author: Lea Schönberger (e-mail: lea.schoenberger@tu-dortmund.de).

This work was partially supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876, project A2, by the Technische Universität München – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement nº 291763, and by the European Commission through the H2020 nIoVe project under Grant Agreement No. 833742.

ABSTRACT Precision time protocol (PTP) is one of the most widely used protocols for clock synchronization in packet-switched networks, on which, among others, the transaction synchronization of the stock markets relies. PTP was not standardized with security as a core requirement and is therefore vulnerable and attractive to manifold kinds of malicious attacks, such as time-delay attacks (TDAs). TDAs, in short, corrupt the exchange of timestamped messages and thus cause an incorrect synchronization process. The annex P of the IEEE 1588-2019 standard has defined a number of security mechanisms for clock synchronization, but, however, none of these can protect a PTP-based system completely against TDAs. In this work, we enhance existing approaches by introducing a so-called observation task and analytically deriving attack properties of an ongoing TDA. Following the recommendation of the annex P of the IEEE 1588-2019 standard, these attack parameters are intended to serve as an additional input for intrusion detection systems to allow for a more reliable and sensitive detection of TDAs. The impact of the derived attack parameters is explored by means of comprehensive experiments.

INDEX TERMS Precision time protocol, real-time, response time analysis, security, time-delay attack

I. INTRODUCTION

PRECISION time protocol (PTP) is one of the most widely used protocols for clock synchronization in packet-switched networks and is based on the exchange of timestamped messages between clients and servers. Among others, it is adopted to synchronize transactions at the stock markets. In 2013, Eurex (one of the world’s largest derivatives exchanges) was halted for a few hours due to an internal time synchronization issue [1], which led to a major loss of profit for many investors. Although this incident was caused by an internal fault, its profound and harmful consequences point out the vulnerability of PTP, turning it into an attractive target for malicious attacks.

In fact, PTP was not standardized with security as a core requirement and therefore is vulnerable to manifold kinds of attacks. One of these is the so-called time-delay attack (TDA) [1]-[3], in the course of which – in short – an attacker, who gained access to the network, delays individual PTP-related messages, which leads to incorrect timestamps being used in the synchronization process. This can have various effects on the network and its participants, as we will discuss later on.

Although the annex P of the IEEE 1588-2019 standard [3] has defined a number of security mechanisms for clock synchronization, none of these measures can protect a PTP-based system completely against TDAs. For this reason, prong D of the annex P of the IEEE 1588-2019 standard does not only emphasize the importance of combining multiple security mechanisms, but specifically recommends to enhance the

---

[1] https://www.reuters.com/article/idINL6N0GR0TY20130826
security of a PTP-based network by monitoring a broad range of parameters. Against this background, we complement existing approaches by making use of response time analysis techniques from the real-time domain and focusing on a parameter, which has, to the best of our knowledge, not yet been considered. By analytically deriving attack properties from the response time of a so-called observation task, we provide an additional input for intrusion detection systems, aiming to allow for a more reliable and sensitive detection of time-delay attacks.

In short, we make the following contributions:

- We introduce an observation task, which can be inserted into existing PTP-synchronized client systems and allows to indicate if a TDA is performed in the network, based on its response time.
- We analytically derive convergence properties of the busy window response time analysis method and exploit these to approximate the attack properties of an ongoing TDA, which can serve as an input to sophisticated intrusion detection systems and thus lead to an increased attack detection sensitivity.
- By means of comprehensive simulations, we explore the impact of the attack parameters of a TDA on the system.

The rest of this work is structured as follows: In Sec. III we explain the precision time protocol (PTP) in detail, before we introduce the threat model adopted in this work in Sec. III-B. In the course of this, we first clarify how time-delay attacks can be performed on PTP in Sec. III-A and, thereon, reveal our attacker model in Sec. III-B. In Sec. IV we provide an overview about existing security and attack mitigation mechanisms with respect to PTP, motivating the novelty and necessity of our analysis proposed in Sec. V. We explore the impact of our derived attack parameters experimentally in Sec. VI before we summarize our findings and conclude this work in Sec. VII.

II. PRECISION TIME PROTOCOL

Precision Time Protocol (PTP) is used for clock synchronization between different participants in a packet-switched network. In networks based on PTP, one participant’s clock serves as the main clock (subsequently termed server clock) with respect to which all other participants’ clocks need to be periodically adjusted. The synchronization process between the server and another network participant (henceforth termed client) is based on the exchange of timestamped messages, by means of which the clock offset, i.e., the difference between the server clock and the client clock, is computed. In the course of this, it is necessary to also determine the transmission delay, which is not part of the clock offset and therefore must be taken into consideration during the offset computation to ensure the correctness of the clock adjustment.

In Fig. 1 a simplified network is portrayed, by means of which the synchronization process is explained in more detail: Initially, the server sends a Sync message to the client and timestamps the moment $t_1$, in which the message leaves the server. Depending on the implementation, this timestamp is transmitted to the client either via the Sync message or by using a so-called Follow_Up message. The client, in turn, timestamps the instant $t_2$ in which it receives the Sync message according to its local clock. Thereon, it computes the difference between $t_2$ and $t_1$, which corresponds to the offset between its local clock and the server clock in addition to the transmission delay. To calculate the transmission delay, the client needs to measure the round-trip time to the server and back. For this purpose, it sends a Delay_Req message to the server and timestamps the moment $t_3$, in which the message leaves the client. Once a Delay_Resp message is received, which is sent by the server and contains the timestamp $t_4$ marking the arrival of the Delay_Req message at the server, the client extracts $t_4$ and proceeds with the computation.

Based on the four timestamps, i.e., $t_1$, $t_2$, $t_3$, and $t_4$, the client can estimate the transmission delay under the assumption that the transmission channel is symmetric, i.e., that the time required for transmitting a message from the server to the client equals the time required for transmitting a message from the client to the server, as given by (1).

$$\text{transmission delay} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \quad (1)$$

After having computed the transmission delay, the clock offset can be calculated as given by (2) and the client’s clock can be updated accordingly.

$$\text{offset} = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \quad (2)$$

III. THREAT MODEL

In the following, we clarify the threat model considered in this work. First, we explain the principle of time-delay attacks on PTP in Sec. III-A before we introduce the adopted attacker model in Sec. III-B.

A. TIME-DELAY ATTACKS ON PTP

As stated in Sec. III the computation of the offset between a server clock and a client clock relies on the assumption that

Please note that this assumption is made according to the section 6.3.3 in the IEEE 1588-2019 standard.

FIGURE 1: A simple network, in which PTP synchronization messages are exchanged. Based on [4].
the communication channel is symmetric, which originates from the IEEE 1588-2019 standard [4] and constitutes a vulnerability attackers can exploit. By analyzing the network traffic, intercepting PTP messages, and artificially retaining (and thus delaying) Sync and/or Delay_Req messages, incorrect timestamps are retrieved by a client, which consequently computes a wrong clock offset and updates its clock incorrectly, as illustrated in Fig. 2. For clarification, consider the offset computation process shown in Fig. 2. Initially, the client clock has an offset of +1 time units, which will be derived correctly if no TDA is performed. However, in the event of a TDA, the client will compute an incorrect offset of +0.5 time units. Please note that a late release instant of the Delay_Req message does not have an impact on the offset computation (which considers only \( t_4 \) and \( t_5 \)), unless the Delay_Req message itself is affected by a time-delay attack.

A TDA can be performed in different ways, namely, it is possible to delay either all messages or to delay PTP messages only. Moreover, a TDA can be performed either in one direction of the communication, known as asymmetric link delay attack [5], or in both directions with different delay [4], termed symmetric delay attack. However, for the remainder of this work, the TDA type is not relevant.

**B. ATTACKER MODEL**

For the rest of this work, we assume that the attacker is either an external entity or an insider with malicious intents [6]. We adopt an in-band adversary model [3], according to which the attacker has full control of the communication path between the server and its clients. However, since we assume the MACsec or IPsec protocol to be used for securing the communication between the server and its clients, inserting fake messages and changing fields of passing messages is detectable and therefore not an option. Instead, the attacker is supposed to aim at compromising the time synchronization of the system using time-delay attacks.

The server is assumed to be trustworthy, such that it cannot be compromised. Moreover, clients are assumed to be secure, so that the attacker cannot compromise them directly. Nevertheless, the attacker is expected to have the expertise and the tools to compromise a switch [4] and thus to be able to compromise all clients indirectly by performing time-delay attacks. Concretely, we assume that a man-in-the-middle attack is implemented in the course of which a TDA on PTP synchronization tasks is performed (as explained in [7]), which introduces asymmetric communication delays, leading to a change of the activation periods of message transmissions.

**IV. RELATED WORK**

Owing to the vulnerability of PTP with respect to security attacks, the challenge of securing PTP and providing detection as well as mitigation mechanisms has been extensively studied [3], [8].

First, the annex K of the IEEE 1588-2008 standard [9] introduced a number of mechanisms to improve the security of PTP. These were extended and enhanced in the latest, recently released version (i.e., in the annex P of IEEE 1588-2019 [4]) and include the deployment of security protocols such as MACsec [10] and IPsec [11] in order to ensure data origin authentication, communication confidentiality, data integrity, as well as replay attack protection and, in consequence, to prevent PTP message manipulation [8], message dropping and insertion [12], denial of service (DoS) attacks [13], as well as master node falsification attacks [14].

With respect to time-delay attacks, multiple detection and mitigation strategies have been proposed in the literature, including architectural mechanisms such as the usage of multiple paths between the server and client clocks [1] or of multiple time sources, i.e., servers [4]. However, although such strategies are theoretically sound, they are not always practical due to their strong dependence on the network topology.

Another direction followed by the research community is to analyze the impact of TDA on the calculation of the round-trip delay and offset at client nodes [2], [13] in order to develop better mitigation mechanisms. To detect a TDA, it is necessary to find inconsistencies in the timestamps of PTP messages.

**FIGURE 2:** Timing effects of asymmetric delaying attacks on PTP messages.
is possible to use a predefined round-trip delay threshold \( [1, 15] \). Whenever this threshold is exceeded, the client assumes that a TDA is in progress. Using this mechanism is, however, not always practical; for instance, if network delays are unbounded, false alarms may be triggered frequently \([1]\). Moreover, if an attacker permanently injects delays larger than the predefined threshold, a denial of service (DoS) attack may be introduced \([2]\).

Apart from these works, much effort has been made to detect anomalies in general based on the timing behavior of a system. In \([16]\), the worst-case execution time (WCET) of each code snippet of a task, as derived based on static timing analysis, is used to detect the execution of unauthorized code. More precisely, the system collects the task’s timing metrics and compares them with predefined worst-case bounds. By doing this, the system becomes capable of detecting situations, in which the observed task is going to exceed its timing requirements due to a security breach. Instead of monitoring at the software level, the usage of dedicated hardware for monitoring the execution of code snippets is proposed in \([17]\). The monitored timing values are compared with precomputed WCET bounds, which allows to detect deviations. Applying worst-case response time analysis for deriving a new bound on the response time of a task and using this as an indicator of code injection attacks has been suggested in \([18]\).

To the best of our knowledge, response-time analysis-based techniques have not yet been applied with regard to the detection of TDAs on PTP. Against this background, we follow the recommendation of the prong D of the annex P of the IEEE 1588-2019 standard \([4]\), according to which the security of a PTP-based network should be enhanced by, in addition to other measures, monitoring a broad range of parameters in the network. Specifically, we subsequently derive attack properties from the response time of an observation task, which can serve as an additional input for intrusion detection systems.

V. TIMING ANALYSIS

As discussed in Sec. \([IV]\) it is meaningful to monitor a large number of parameters in order to detect time-delay attacks and to be able to invoke countermeasures. Although the number of parameters that can be monitored is large, not all of them are useful in this context. For instance, when considering a client system on which one or more tasks are delayed due to a TDA in the network (cf. Sec. \([III]\), monitoring the execution time of tasks is not meaningful for detecting the attack. In fact, the events of the respective tasks are delayed, but none is skipped, for which reason the TDA does not have any impact on their execution times. What, in contrast, is affected during a TDA is the response time of tasks, as illustrated in the example in Fig. \([3]\). If the activation of the higher-priority task \( \tau_j \) is delayed by \( \delta \) time units (cf. second diagram in Fig. \([3]\)), the response time of the lower-priority task \( \tau_k \) is shorter compared to the case in which the activation of \( \tau_j \) is not delayed (cf. first diagram in Fig. \([3]\)). Accordingly, observing the response time of tasks on a client system appears to be a promising approach in order to detect TDAs in the network.

However, monitoring the response times of all (PTP-synchronized) tasks on all client systems in a network is inconvenient. For this reason, we subsequently introduce so-called observation tasks, which are scheduled under the lowest priority on each client system (strategies for integrating an observation task into an existing system can, e.g., be found in \([19]\)). The response time of an observation task includes the interference from all higher-priority tasks and therefore reflects the effects of delays following from a TDA. Instead of simply comparing the observed response time of an observation task with the so-called nominal worst-case response time, i.e., the worst-case response time in the case that no TDA is performed in the network, we go one step further in this work and derive attack parameters from the observed response time, which can be used by intrusion detection systems to not only detect a TDA, but also to invoke countermeasures.

To this end, we subsequently provide a system model in Sec. \([V-A]\) and model the timing behavior of the system in Sec. \([V-B]\) before we revisit the theoretical foundations of the busy window response time analysis in Sec. \([V-C]\) and derive convergence properties, which are exploited in Sec. \([V-D]\) to obtain approximations of the attack parameters of a TDA. In Sec. \([V-E]\) we provide some additional remarks. A supportive quick reference for the notation is given in Table \([I]\).

---

\(*\) Please note that monitoring the execution times of tasks may be meaningful to detect other attacks such as code injection attacks, where additional instructions are inserted by an attacker. However, this is beyond the scope of this work.

---

\[\text{FIGURE 3: An example illustrating the effect of a delayed activation of the higher-priority task } \tau_j \text{ on the response time of the lower-priority task } \tau_k \text{ (second diagram) in comparison to a case in which the activation of } \tau_j \text{ is not delayed (first diagram).}\]
A. SYSTEM MODEL

For the rest of this work, we consider a client system in a PTP-based network, on which a set of sporadic tasks $T$ is assumed to be executed under a preemptive fixed-priority scheduling policy according to a given priority assignment. Each task $\tau_i \in T$ is characterized by its worst-case execution time (WCET) $C_i$ and its period (or inter-arrival time) $P_i$. Moreover, $\alpha_i$ describes the nominal event arrival function of a task $\tau_i$, providing an upper bound on the number of resource accesses per period of time. For each task $\tau_i$, the set of interferers, i.e., the set of higher-priority tasks, is denoted as $I_i$. The activation of (a subset of) tasks is synchronized using PTP synchronization operations (cf. Sec. III-B) who is able to intercept and thus to analyze the network traffic.

| Notation | Explanation |
|----------|-------------|
| $\alpha_i$ | nominal event arrival function (upper bound) of task $\tau_i$ |
| $\delta_i$ | off-nominal event arrival function (upper bound) of task $\tau_i$ |
| $C_i$ | worst-case execution time (WCET) of task $\tau_i$ |
| $d_i$ | number of delayed events of task $\tau_i$ due to a TDA |
| $\delta$ | fixed periodic delay resulting from a TDA |
| $\delta^{\text{max}}$ | minimum possible periodic delay attack |
| $\delta^{\text{max}}$ | maximum possible periodic delay without causing a DoS |
| $f_i$ | computation function of the busy window of task $\tau_i$ |
| $f_i(n)$ | computation function of the busy window of task $\tau_i$ at iteration $n$ |
| $F_i(n)$ | bound on the computation function of the busy window of task $\tau_i$ |
| $q_i$ | q-event earliest possible activation of task $\tau_i$ |
| $I_i$ | set of interferers of task $\tau_i$ |
| $I^\text{set-affected}$ | set of interferers of task $\tau_i$ which are not affected by a TDA |
| $I^\text{set-afflicted}$ | set of interferers of task $\tau_i$ which are affected by a TDA |
| $l_i$ | number of periodic events of task $\tau_i$ after which one event is delayed |
| $l^{\text{min}}$ | minimum possible value of $l_i$ for task $\tau_i$ |
| $l^{\text{max}}$ | maximum possible value of $l_i$ for task $\tau_i$ |
| $\ell_i$ | convergence iteration of the computation function of the busy window of a task |
| $P_i$ | period of a task $\tau_i$ |
| $R_i$ | worst-case response time (WCRT) of task $\tau_i$ |
| $\rho_i$ | rate of convergence of the computation function of the busy window of task $\tau_i$ |
| $T$ | set of tasks on the contemplated client |
| $\tau_1$ | task; general notion |
| $\tau_j$ | interfering task of $\tau_k$ |
| $\tau_k$ | observation task with lowest priority |
| $w_i(q)$ | q-event busy window of task $\tau_i$ |
| $\Omega_i$ | l-event busy window of task $\tau_i$ |
| $\Omega^{\text{set}}$ | size of the l-event busy window of task $\tau_i$ at iteration $n$ |
| $\Omega^{\text{overall}}$ | overall delay introduced to the client system due to a TDA |

B. TIMING AND EVENT MODEL

In order to analyze the response time of an observation task $\tau_k$, it is necessary to first define the previously mentioned nominal worst-case response time of a task as well as the method by means of which it is computed. Concretely, the observation task is analyzed during a busy window $w_k(q)$, which is defined as follows:

**Definition 1** (Busy Window). Let the busy window $w_k(q)$ be the maximum amount of time required to complete $q$ activations of task $\tau_k$. It is bounded by

$$w_k(q) \leq q \cdot C_k + \sum_{j \in I_k} \alpha_j(w_k(q)) \cdot C_j.$$  \hspace{1cm} (3)

Based on Def. I the nominal worst-case response time $R_k$ of task $\tau_k$ is defined as follows:

**Definition 2** (Nominal Worst-Case Response Time). The nominal worst-case response time $R_k$ of a task $\tau_k$ is defined as

$$R_k = \max\{w_k(q) - \phi_k(q) | \phi_k(q + 1) \leq w_k(q)\}.$$ \hspace{1cm} (4)

It is evident from Def. I that $w_k(q)$ of the analyzed task $\tau_k$ strongly depends on the nominal event arrival functions $\alpha_j$ of its interferers $\tau_j \in I_k$. However, in the case that a TDA is performed in the network, a delay is introduced to a number of tasks in $I_k$, as described above, which leads to a deviation from their nominal event arrival functions. To model the effect of a TDA on a task $\tau_j \in I_k$, an off-nominal event arrival function $\tilde{\alpha}_j$ over an interval of time $\Delta t$ can be defined, for which holds that

$$\tilde{\alpha}_j(\Delta t) \leq \alpha_j(\Delta t) \quad \forall \Delta t.$$ \hspace{1cm} (5)

According to the threat model provided in Sec. III we subsequently assume that due to the delayed activation of a task $\tau_j$ every $(\ell_j + 1)^{th}$ event within the attack window is delayed by $\delta$ time units, as illustrated in Fig. A. With respect to all non-delayed events, i.e., where $\mod(\ell_j + 1) \neq 0$, the minimum distance between two subsequent events remains unchanged and thus equals the task period $P_j$. Accordingly, the off-nominal event arrival function $\tilde{\alpha}_j$ of a delayed task $\tau_j$ can be defined as follows:

**Definition 3** (Off-Nominal Event Arrival Function). The off-nominal event arrival function of a task $\tau_j$ is defined as the sporadically periodic event function...
For clarification, consider each complete sequence of \( \ell_j \) events, followed by a time interval of \( P_j + \delta \) as a section of the interval \( \Delta t \): The first part of Eq. (6) counts the number of events contained in all complete sections in \( \Delta t \), while the second part of Eq. (6) factors in the remaining events. Please note that the number of events within \( \Delta t \) computed by \( \alpha_j \) is an upper bound.

Being able to quantify the events of a delayed task using the off-nominal event arrival function given in Def. 3, it is necessary to redefine the busy window introduced in Def. 1 in order to include the impact of the off-nominal interference of one or more delayed \( \tau_j \) in the response time of the observation task \( \tau_k \).

**Definition 4 (Busy Window Revisited).** Let the busy window \( w_k(q) \) be the maximum time required to complete \( q \) activations of task \( \tau_k \). It is bounded by

\[
w_k(q) \leq q \cdot C_k + \sum_{\tau_j \in I_k^{\text{da}}} \alpha_j(w_k(q)) \cdot C_j + \sum_{\tau_j \in I_k^{\text{non}}} \alpha_j(w_k(q)) \cdot C_j
\]

where \( I_k^{\text{da}} \) is the set of interferers of \( \tau_k \) delayed due to a TDA in the network and \( I_k^{\text{non}} \) is the set of non-delayed interferers, for which holds that \( I_k^{\text{da}} \cup I_k^{\text{non}} = I_k \) and \( I_k^{\text{da}} \cap I_k^{\text{non}} = \emptyset \).

Please note that Def. 4 boils down to Def. 1 if no TDA is performed in the network, i.e., if \( I_k^{\text{da}} = \emptyset \).

So far, the nominal event arrival function of tasks as well as the nominal worst-case response time of the observation task \( \tau_k \) have been defined, allowing us to compare the latter to the actual response time of \( \tau_k \) in order to identify deviations. However, we are interested in retrieving more information about a TDA based on such deviations, namely, the so-called attack parameters, i.e., approximations of the delay \( \delta \), the overall delay \( \Omega \) introduced to the client system, and the number of events \( \ell_j \) after which one event of a task \( \tau_j \) is delayed. Since the off-nominal event arrival functions \( \alpha_j \) of delayed tasks \( \tau_j \in I_k \), which depend on \( \delta \) and \( \ell_j \) (cf. Def. 3), are unknown, it is necessary to bridge the gap between these and the actual response time of the observation task \( \tau_k \).

In the following, we dive deeper into the theory of the busy-window method and derive convergence properties, that can be exploited for this purpose.

**C. Convergence Properties of the Busy-Window Method**

As evident from Def. 4, the busy window is calculated by a fixed-point computation, which is solved iteratively in a number of discrete steps. To facilitate the readability, we subsequently consider the computation of a one-event busy window, i.e., \( q = 1 \), for a task \( \tau_i \). In this contemplated case, the computation starts at iteration 0, where \( w_i^0 = C_i \), i.e., considering only the WCET of \( \tau_i \) without any interference. Thereon, one of the interferers \( I_i \) is added in each iteration, until the computation reaches a fixed point. This process is illustrated on the left side of Fig. 5.

Let \( f_i(n) = w_i^n \) be a discrete function associating to each iteration \( n \) a value \( w_i^n \) describing the size of the busy window. This function \( f_i \) is referred to as the computation function of the busy window and is illustrated on the right side of Fig. 5.

Although the computation function \( f_i \) is not known a priori, a bound can be obtained using numerical analysis. To this end, we make the following assumptions:

(a) The event arrival functions of all \( \tau_i \in T \) are sub-additive, preventing the size of the busy window from growing asymptotically.

(b) When the workload of no not already considered \( \tau_j \in I_i \) can be added from one iteration to the next one, the busy window converges and its size reaches a fixed point of value \( w_i^n \). That is,

\[
w_i^{n+1} - w_i^n = 0 \iff \sum_{j \in I_i} \alpha_j(w_i^n) - \sum_{j \in I_i} \alpha_j(w_i^{n-1}) = 0 \quad (8)
\]

Aiming to bound the computation function \( f_i \) of the busy window of a task \( \tau_i \), it is necessary to reflect upon its rate of convergence. In general, the rate of convergence \( f_i(n+1) - f_i(n) \) determines the ratio between the error at iteration \( n + 1 \) and the error at the previous iteration. Since \( r \) is not known in advance, the error \( f_i(n) - r \) at iteration \( n \) is approximated with the value \( f_i(n) - f_i(n-1) \).

For the specific \( f_i(n) = w_i^n \) considered in this work, the following lemma can be formulated:

**Lemma 1 (Rate of Convergence).** For the rate of convergence \( \rho_i \) of the computation function \( f_i \) of the busy window of a task \( \tau_i \), it holds that

\[ f_i(n+1) - f_i(n) \leq \rho_i \]

An error at an iteration \( n \) refers the difference between the value \( f_i(n) \), i.e., at iteration \( n \), and the value of the fixed point \( r \).
Thus, computation function \( f \),

\[
\sum \sum w \rho f \quad \text{F}
\]

The difference between two computation iterations bounded by the function \((\text{Bound on the Computation Function})\) and the busy window at the previous iteration account the rate of convergence, which determines the error \( n \).

\[
\sum w \tau j C \quad VOLUME 4, 2016
\]

\[
\rho_i = \frac{w^{n+1} - w^n}{w_i^n - w_{i-1}^n} \leq \sum C_j P_j \leq 1. \quad (9)
\]

**Proof.** In order to determine the rate of convergence \( \rho_i \), it is necessary to consider the difference of the size of the busy window between two iterations \( w_i^{n+1} \) and \( w_i^n \). According to Def. [1] for a 1-event busy window, \( w_i^{n+1} = C_i + \sum_{j \in I_i} \omega_j(w_i^n) \cdot C_j \) and \( w_i^n = C_i + \sum_{j \in I_i} \omega_j(w_i^{n-1}) \cdot C_j \). Therefore, \( w_i^{n+1} - w_i^n = \sum_{j \in I_i} \omega_j(w_i^n) \cdot C_j - \sum_{j \in I_i} \omega_j(w_i^{n-1}) \cdot C_j \leq \sum_{j \in I_i} \left( \frac{w_i^n + 1}{p_j} \right) \cdot C_j - \sum_{j \in I_i} \left( \frac{w_i^{n-1} + 1}{p_j} \right) \cdot C_j = \sum_{j \in I_i} w_i^{n+1} - w_i^n \leq \left( \frac{w_i^n}{p_j} \cdot C_j \right) \cdot \sum_{j \in I_i} p_j \leq 1. \quad \Box

Based on the rate of convergence of the computation function \( f_i \) of the busy window, it is possible to determine a bound on \( f_i \):

**Theorem 1 (Bound on the Computation Function).** The computation function \( f_i(n) \) of the busy window of a task \( \tau_i \) is bounded by the function

\[
F_i(n) = \rho_i^n \cdot (w_i^1 - w_i^0). \quad (10)
\]

**Proof.** The difference between two computation iterations of the busy window is defined as \( F_i(n) = w_i^{n+1} - w_i^n \) at iteration \( n \). It can also be defined based on the value of the busy window at the previous iteration \( n - 1 \), taking into account the rate of convergence, which determines the error between the two iterations, i.e., \( F_i(n) = \rho_i \cdot (w_i^n - w_i^{n-1}) \). This can be further repeated until the initial iteration, taking into account the error by considering the rate of convergence \( \rho_i \) at each iteration.

In what follows, we exploit the above derived convergence properties of the busy window method to retrieve approximations of the attack parameters of a TDA based on the response time of an observation task \( \tau_k \).

**D. APPROXIMATION OF THE ATTACK PARAMETERS**

As explained in Sec. [V-B] the response time of an observation task \( \tau_k \) does not equal its nominal worst-case response time if a TDA is performed in the network, but the size of the busy window is reduced due to the delayed events of a (number of) higher-priority task(s) \( \tau_j \in I_k \), as stated in Def. [4] We refer to this reduced busy window by \( w_k^{tda} \) and assume that \( w_k^{tda} < w_k \), where \( w_k \) corresponds to the nominal WCRT of \( \tau_k \). Moreover, we assume that the computation function \( f_k \) of the busy window converges at iteration \( n_{tda} \) – which is unknown, but can be derived as follows:

**Theorem 2 (Convergence Iteration).** The iteration \( n_{tda} \) at which the computation function \( f_k \) of the busy interval of a task \( \tau_k \) converges and the size of the busy window reaches \( w_k^{tda} \) is given by

\[
n_{tda} = \begin{cases} 
\log_{\rho_k} \left( 1 - \frac{(w_k^{tda} - C_k)(1 - \rho_k)}{w_k^{tda} - w_k} \right) - 1 & \text{if } \rho_k < 1 \\
\frac{w_k^{tda} - C_k}{w_k^{tda} - w_k} - 1 & \text{if } \rho_k = 1
\end{cases}
\]

(11)

where \( w_k^{tda} \) indicates the observed response time of \( \tau_k \).

**Proof.** The value \( w_k^{tda} \) can be written as \( w_k^{tda} = C_k + \sum_{n=0}^{n_{tda}} F_k(n) \), which equals the value of \( w_k^0 \) at the first iteration plus the accumulated difference of the added interference between two successive iterations until iteration \( n_{tda} \). Using Theorem [1] \( w_k^{tda} = C_k + \sum_{n=0}^{n_{tda}} \rho_k^n \cdot \left( w_k^1 - w_k^n \right) \), \( C_k = \left( w_k^1 - w_k^0 \right) \cdot \sum_{n=0}^{n_{tda}} \rho_k^n \). The term \( \sum_{n=0}^{n_{tda}} \rho_k^n \) is a geometric series, for which two cases must be distinguished,
Theorem 3 (Overall Delay and Delayed Events). The overall delay \( \Omega \) introduced to the client system due to a TDA delaying one interferer \( \tau_j \in I_k \) is given by

\[
\Omega = f_k(n_{tda} + 1) - f_k(n_{tda})
\]  

(12)

and the number of delayed events \( \tilde{q}_j \) of \( \tau_j \) by

\[
\tilde{q}_j = \left\lceil \frac{\Omega}{P_j} \right\rceil.
\]  

(13)

Proof. Since we already discussed the properties of the convergence function \( f_k \) of the busy interval in Sec. V-C we only sketch the proof. As illustrated in Fig. 6 it holds by construction that \( f_k(n_{tda} + 1) = f_k(n_{tda}) + \Omega \). Following from the assumption that only one \( \tau_j \in I_k \) is delayed in consequence of a TDA, only events of \( \tau_j \) can have been delayed during \( \Omega \). Therefore, the number of delayed events of \( \tau_j \) is upper-bounded by \( \tilde{q}_j = \left\lceil \frac{\Omega}{P_j} \right\rceil \).

Having considered the case that only one \( \tau_j \in I_k \) is delayed in consequence of a TDA, we henceforth assume that multiple \( \tau_j \in I_k \) are delayed. While this assumption does not have any impact on the overall delay \( \Omega \), no definite statement about the number of delayed events per task can be made in this case. Since it is unknown, which task contributed how much to \( \Omega \), and since finding a distribution of \( \Omega \) that corresponds to the real system state is not trivial, we suggest to approximate it by means of a heuristic.

As a straightforward approach to approximate the number of delayed events per task under the assumption that the number of delayed tasks is known, it is possible to distribute \( \Omega \) according to a uniform distribution. However, since not all tasks contribute the same amount of workload to the busy window \( w_k^{tda} \), it is more sensible to distribute \( \Omega \) proportionally to the task utilization. For clarification, consider two tasks \( \tau_j, \tau_y \in I_k \) delayed in consequence of a TDA, to which, proportionally to their utilization, i.e., \( C_j \) and \( C_y \), respectively, the following shares of \( \Omega \) are distributed: \( s_j \cdot \Omega \) to \( \tau_j \) and \( s_y \cdot \Omega \) to \( \tau_y \), where \( s_j + s_y = 1 \). Based on this distribution, the number of delayed events per task can be approximated similarly to Theorem 3, i.e., \( \tilde{q}_j = \frac{s_j \cdot \Omega}{P_j} \) and \( d_y = \frac{s_y \cdot \Omega}{P_y} \). If the number of delayed tasks, however, is unknown, all possible combinations of potentially delayed tasks must be enumerated and the approximated number of delayed events for each enumerated scenario must be computed in order to cover all possible system states.

Although the number of delayed events per task can be determined (in case only one task is delayed) or at least approximated and enumerated (in case multiple tasks are delayed), determining the exact periodic delay \( \delta \) is not possible, since not only \( \delta \), but also \( \ell_j \) and, in consequence, \( \tilde{q}_j \) are unknown. Nevertheless, bounds on \( \delta \) can be provided: As already known from Sec. V-A, \( \delta \) is upper-bounded by \( \delta^{max} \). To determine a lower bound \( \delta^{min} \), we again first consider the case that only one \( \tau_j \in I_k \) is affected by a TDA. Since in this case the number of delayed events can be computed using Theorem 3, \( \delta^{min} \) is obtained by:

Theorem 4 (Minimum Periodic Delay). The minimum periodic delay \( \delta^{min} \) that can have been introduced to a PTP-synchronized interferer \( \tau_j \in I_k \) of the observation task \( \tau_k \) due to a TDA is given by

\[
\delta^{min} = \frac{\Omega}{\alpha_j(w_k) - \tilde{q}_j}.
\]  

(14)

Proof. Since the number of events of task \( \tau_j \) during the nominal busy window \( w_k \) of the observation task \( \tau_k \) can be determined by its well-known nominal event arrival function \( \alpha_j(w_k) \), the number of events of task \( \tau_j \) during the off-nominal busy window \( w_k^{tda} \) of \( \tau_k \) can be computed as \( \alpha_j(w_k) - \tilde{q}_j \), where \( \tilde{q}_j \) is the number of delayed events. Although the number of periodic activations of \( \tau_j \) after which one event is delayed by \( \delta \) time units is unknown, the minimum possible value of \( \ell_j \) according to the threat and system models is \( \ell_j^{min} = 1 \), which corresponds to the case that each event is delayed. Therefore, \( \delta^{min} \) is obtained by dividing the overall introduced delay \( \Omega \) by the number of events, i.e., \( \delta^{min} = \frac{\Omega}{\alpha_j(w_k) - \tilde{q}_j} \).

The approximation of \( \delta^{min} \) and the knowledge of \( \delta^{max} \) do not allow to determine the exact delay \( \delta \), but can be used to enumerate all possible values of \( \delta \). This enumeration can, for instance, be done by starting from \( \ell_j = \ell_j^{min} = 1 \) and increasing the value of \( \ell_j \) in discrete steps, where the related value of \( \delta \) can be computed similarly to Theorem 4 dividing \( \Omega \) by the number of \( \ell_j \)-event sections. All possible values of \( \delta \) have been enumerated as soon as one computed value of \( \delta \) is larger than \( \delta^{max} \). Then, \( \ell_j^{max} = \ell_j - 1 \), where \( \ell \) is the value used in the recently completed computation. In the case that more than one \( \tau_j \in I_k \) is delayed in consequence of a
FIGURE 6: Busy window computation dynamics of the observation task $\tau_k$ under the overall introduced delay $\Omega$, simplified as an accumulation of all occurrences of $\delta$.

TDA, the same approach can be applied, however, taking all approximations of $\tilde{q}_j$ for all delayed $\tau_j \in I_k$ into account.

E. REMARKS ON THE ANALYSIS
So far, we derived convergence properties from the busy window response time analysis method and used these to retrieve attack parameters of an ongoing TDA. In this context, we append two further remarks:

1) The derived attack parameters are approximations and are not required to be precise. This is particularly the case because they are intended to be fed into an intrusion detection system in addition to many further parameters and, therefore, play a supportive role. More specifically, the intrusion detection system does not rely on the attack parameters, but can exploit these in order to make more sensitive predictions and to invoke more effective countermeasures.

2) It is well-known that in the majority of cases tasks do not execute for their worst-case execution time. However, in our formal analysis, we consider their worst-case execution times anyway. This follows from the fact that the applied busy window response time analysis technique is designed for analyzing the worst-case response time of tasks and therefore relies on the usage of worst-case execution time values. This may introduce a high degree of pessimism into our approach and could lead to an overly sensitive intrusion detection system triggering frequent false alarms, but can be avoided if realistic worst average-case execution time values are provided.

F. REMARKS ON JITTERS
In the following, we will discuss the impact of jitters on the analysis proposed in this work. Jitters are deviations of the event arrival from a task’s period. More precisely, an event can arrive earlier (negative jitter) or later (positive jitter) than the period due to environmental influences such as, e.g., the material characteristics of a hardware component.

A jitter is extremely small compared to a task period and can be bounded by the maximum negative jitter $J^-\tau$ and the maximum positive jitter $J^+\tau$ for a task $\tau_i$. Typically, a jitter is in the interval $[J^-\tau, J^+\tau]$ and occurs at every event release. However, the maximum jitters occur only in very rare cases.

Based on the maximum jitters, an upper and a lower bound on the nominal event arrival function $\alpha_i$ can be defined, namely, the nominal event arrival function with maximum negative jitter $\alpha_i^-$ and the nominal event arrival function with maximum negative jitter $\alpha_i^+$, where the maximum negative (or positive, respectively) jitter occurs in each period.

**Definition 5** (Nominal Event Arrival Functions with Maximum Jitter). The nominal event arrival function with maximum negative jitter of a task $\tau_i$ is defined as

$$\alpha_i^- (\Delta t) = \left[ \frac{\Delta t}{P_i - J^-\tau_i} \right].$$

and the nominal event arrival function with maximum positive jitter of a task $\tau_i$ is defined as

$$\alpha_i^+ (\Delta t) = \left[ \frac{\Delta t}{P_i + J^+\tau_i} \right].$$

Similarly, an off-nominal event arrival function with maximum negative jitter $\tilde{\alpha}_i^-$ and with maximum positive jitter $\tilde{\alpha}_i^+$ can be defined on

Consider the case that under a no-attack scenario each event of a task $\tau_j$ arrives with the maximum positive jitter $J^+\tau_j$, as illustrated on the left side of Fig. 7. In consequence, the accumulation of jitters may at some point in time equal the length of one period, such that until this point in time one event less has arrived compared to the zero-jitter case. Accordingly, the accumulated jitter will have an impact on

---

8We omit the definitions at this point.
9Please note that the scenario is unrealistically dramatized in Fig. 7 to allow for a more readable illustration.
the response time of the observation task and, therefore, is likely to be considered as a delay introduced by a TDA. However, such a scenario can only occur if the jitter is large enough compared to the period and, moreover, if the observation interval is long enough.

Further, consider the case that a TDA is performed, resulting in an off-nominal event arrival function $\tilde{\alpha}_j$ of task $\tau_j$ including a constant delay $\delta$, as illustrated on the right side of Fig. 7. Additionally, assume that each event of $\tau_j$ arrives with a maximum negative delay $J^-_j$. In this scenario, the accumulated $J^-_j$ may have amortized the delay $\delta$ at some point in time, such that the attack does not have any impact on the response time of the observation task. However, similar to the previous case, this can only occur if the jitter is large enough compared to the period, if the observation interval is long enough, and, moreover, if $\delta$ is small enough compared to the jitter and if the sequence of undelayed events before each introduced $\delta$ is long enough.

Both of the discussed cases are extremely rare, since the probability of event arrivals according to the nominal event arrival function with maximum positive or negative jitter is infinitesimal, whereas the typical jitter does not have a significant impact on the proposed analysis.

VI. EVALUATION

To study the impact of the attack parameters of a TDA on the response time of an observation task, we first explain how to model a PTP-based network for compositional performance analysis (CPA), before we perform two experiments using pyCPA [21], namely, one based on synthetic task sets in Sec. VI-B and one based on an exemplary network with in Sec. VI-C

A. MODELING FOR CPA

Fig. 8 illustrates a simplified compositional performance analysis (CPA) model of a PTP-based network, including a server node, a client node, and a PTP-aware switch. The output ports of the switch are mapped to CPA resources using a fixed-priority non-preemptive scheduling policy. On the switch, four tasks are depicted, which are required for performing the PTP synchronization operations (cf. Sec. II). For each task, it holds that the lower the index, the higher the priority. Please note that in non-real-time operating systems such as Linux, PTP is implemented using one task only (the so-called PTP daemon). The decision to model PTP using four (sub-)tasks aims at modeling the effect of a TDA in a more fine-grained way.

The PTP message flow starts in the event source Periodic $\text{Sync}$ on the server, which generates the periodic activation of $\text{Sync}()$ messages, represented by task $\tau_1$ in the server node. $\text{Sync}()$ messages are propagated through the path, including task $\tau_1$ in the switch, until the input port of the client node, where it activates task $\tau_1$ on the client node. Task $\tau_2$ represents the $\text{Follow\_Up()}$ message sent by the master node. The IEEE 1588-2019 standard requires that $\text{Follow\_Up()}$ is transmitted as early as possible after the transmission of the related $\text{Sync()}$ message [4]. This message is propagated through the model in the same way as described for $\text{Sync()}$. In multi-cast communication, the IEEE 1588-2019 standard requires that the $\text{Del\_Req()}$ message is generated with a particular periodicity. Therefore, the event source (Periodic $\text{Sync}$) on the client node is used to activate task $\tau_3$, which sends the message $\text{Del\_Req()}$ that is propagated through the switch until the input port of the server node. There, it activates the task $\tau_3$, which creates the $\text{Del\_Resp()}$ message to be propagated through the path until task $\tau_4$ in the client node.

In the remainder of this work, we assume PTP to be modeled similarly in pyCPA, making adjustments as required for the respective experiment setups.

B. SYNTHETIC EVALUATION

In our first experiment, we explore the impact of different values of the attack parameters of a TDA on the WCRT of the observation task, applying our proposed analysis to synthetically generated task sets. To this end, we first introduce our experiment setup in Sec. VI-B1 before we discuss the results in Sec. VI-B2.
In our second experiment, we consider exemplary attack scenarios. It is more reasonable to enumerate and consider all possible attack parameters of a TDA. Instead, it is evident that based on one observed response time of $\tau_k$, the lower the value of $\delta_j$, the more frequently a delay is introduced, the lower the response time of $\tau_k$ decreases. Moreover, it is evident that the lower the value of $\ell_j$, i.e., the more frequently a delay is introduced, the lower the response time of $\tau_k$. However, the relations between $\delta$ and the response time of $\tau_k$ as well as between $\ell_j$ and the response time of $\tau_k$ are clearly not linear, but follow the dynamics elucidated in our analysis in Sec. V.

In Fig. 10 different combinations of values of the parameters $\delta$ and $\ell_j$ are presented, which all result in a busy window $w_k^\text{tda} = 255$ of the observation task $\tau_k$. Following from this, it is evident that based on one observed response-time value of $\tau_k$, it is not always possible to make a statement about the particular attack parameters of a TDA. Instead, it is more reasonable to enumerate and consider all possible attack scenarios.

**C. USECASE EVALUATION**

In our second experiment, we consider an exemplary use-case. The experiment setup is explained subsequently in Sec. VI-C1 before we discuss the results in Sec. VI-C2.
TABLE 2: Packet sizes and time required for transmitting the packets, here considered as WCET.

| Message/Packet | Packet Size (Byte) | WCET (µs) |
|----------------|-------------------|-----------|
| Sync           | 106               | 9         |
| Follow_Up      | 106               | 9         |
| Delay_Req      | 106               | 9         |
| Delay_Resp     | 116               | 10        |
| Best Effort Traffic (Video) | 875 - 1400 | 112 |

TABLE 3: Tasks considered in the experimental evaluation.

| Task                  | Period |
|-----------------------|--------|
| Sync (τₗj)            | 61 (P_Sync) |
| Delay_Req             | 106 (P_Sync) |
| Observation task (τₖj) | 73.2 (1.2 × P_Sync) |

traffic.

When modeling the considered system, we follow a similar approach as proposed by Diemer et al. [21]: The system is modeled as an Ethernet AVB network, in which each client has input and output ports and each switch executes channel-delay tasks representing the default propagation and processing delays. Ports are modeled using a fixed-priority preemptive scheduling policy. Moreover, each output port accounts for the most significant arbitration delays. Static delays such as those caused by propagation in the communication channel are incorporated as constant-delay overhead in the paths. The analysis is performed using pyCPA [21].

Since the considered system is assumed to use an Ethernet 100BASE-TX network (100 Mbps), for each PTP synchronization message transmitted on the network, the payload is assumed to comply with the respective number of bits indicated by the IEEE 1588 standard [4], excluding additional bytes, i.e., the Ethernet header, the MACsec header, and the octets annexed at the physical link. The size of each video packet is assumed to range from 875 bytes to 1400 bytes.

Table II summarizes the total length of the PTP and video packets as well as the respective WCET, i.e., here, the time required for transmitting the packets at 100 Mbps. The timing characteristics of the PTP-related and observation tasks running on the client are given in Table III.

2) Results

In Fig. 12 the impact of the attack parameters δ and ℓₗj of a delayed task τₗj is depicted for different values of ℓₗj with δ being varied in discrete steps within the interval [0, 12]µs. It can be seen that the response time of the observation task τₖj is reduced from 112 µs to 104 µs for ℓₗj = 1, ℓₗj = 2, and ℓₗj = 3 as soon as δ ≥ 2µs (δ ≥ 3µs, respectively). However, no change is detectable for the same value of δ if ℓₗj = 4. Concretely, this means, that a less frequently introduced delay has no impact in this case, whereas a more frequently introduced delay of the same length does. A corresponding behavior has already been observed in Fig. 9 in Sec. VI-B2

where, e.g., for ℓₗj = 2 at time 20 the response time of the observation task τₖj is much lower than for ℓₗj = 16.

Moreover, Fig. 12 gives the hint that the choice of the worst-case execution time of the observation task τₖj should be made very carefully. Depending on the task parameters of a specific usecase, the level of detail retrievable by means of the response time of the observation task can differ largely based on the considered parameters of the response time. Therefore, more fine-grained observations may be made by using different types of observation tasks. Exploring this, however, is beyond the scope of this work.

VII. CONCLUSION

Aiming to contribute to the development of more reliable and sensitive intrusion detection systems for PTP-based networks, we proposed to introduce observation tasks scheduled under the lowest priority on each client system. We showed, how, based on analysis techniques from the real-time domain, conclusions from the response time of an observation tasks to the existence of a TDA in the network can be drawn. Moreover, we analytically derived convergence properties of the busy window worst-case response time analysis methods, which do not only provide new theoretical insights, but can also be exploited to approximate attack parameters of an ongoing TDA.

De facto, the attack parameters derived in our analysis can serve as an additional input for sophisticated intrusion detection systems such as, e.g., the so-called Red-Zone Principle [18], allowing them to select better bounds for distinguishing off-nominal from nominal system behavior; thus, also enabling them to detect TDAs more reliably and to invoke mitigation strategies effectively. Optimizing such bounds for a specific intrusion detection system and determining the resulting false positive and false negative rates remains future work.

For more information, please refer to IEEE 802.3-2018 [23], Section 1.
By means of comprehensive simulations, we explored the impact of different configurations of the attack parameters on the response time of an observation task and showed that different configurations can lead to the same response time of an observation task, emphasizing the importance of approximations, since the exact attack parameters cannot always be retrieved. Moreover, we discovered in our usecase simulation that the choice of the execution time of observation tasks is extremely important and has a strong impact on the detection sensitivity of an intrusion detection system.

REFERENCES

[1] T. Mizrahi, “A game theoretic analysis of delay attacks against time synchronization protocols,” in 2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings. IEEE, 2012, pp. 1–6.

[2] M. Ullmann and M. Vögel, “Delay attacks—implication on ntp and ptp time synchronization,” in 2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication. IEEE, 2009, pp. 1–6.

[3] E. Fikin 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, 2017.

[4] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008), pp. 1–499, 2020.

[5] R. Annessi, J. Fabini, F. Iglésias, and T. Zeby, “Encryption is futile: Delay Attacks on High-Precision Clock Synchronization,” CoRR, vol. abs/1811.08569, 2018. [Online]. Available: http://arxiv.org/abs/1811.08569

[6] T. Mizrahi, “Security requirements of time protocols in packet switched networks,” in RFC 7384, 2014.

[7] 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.

[8] J. Tsang and K. Beznosov, “A security analysis of the precise time protocol,” in Information and Communications Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 50–59.

[9] Institute of Electrical and Electronics Engineers, “IEEE Std 1588-2008, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” 2008.

[10] IEEE, “IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006): IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security,” 2018.

[11] S. Kent and R. Atkinson, “Security architecture for the internet protocol,” 1998.

[12] G. Gaderer, A. Treytl, and T. Sauter, “Security aspects for IEEE 1588 based clock synchronization protocols,” in IEEE International Workshop on Factory Communication Systems (WFCS06), Torino, Italy. Citeseer, 2006, pp. 247–250.

[13] M. Han and P. Crossley, “Vulnerability of IEEE 1588 under time synchronization attacks,” in 2019 IEEE Power & Energy Society General Meeting (PESGM). IEEE, 2019, pp. 1–5.

[14] B. Moussa, C. Robillard, A. Zugemainter, M. Kassouf, M. Debbabi, and C. Assi, “Securing the precision time protocol (PTP) against fake timestamps,” IEEE Communications Letters, vol. 23, no. 2, pp. 278–281, 2018.

[15] E. Lisova, M. Gutiérrez, W. Steiner, E. Uhlemann, J. Åkerberg, R. Dobrin, and M. Han, “Time-based intrusion detection in cyber-physical systems,” in Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems, 2010, pp. 109–118.

[16] N. Bellèce, S. Rokicki, and I. Puaut, “Attack detection through monitoring of timing deviations in embedded real-time systems,” in ECRS 2020 – 22nd Euromicro Conference on Real-Time Systems, Modena, Italy, Jul. 2020, pp. 1–22. [Online]. Available: https://hal.inria.fr/hal-02559549

[17] M. Hamad, Z. A. H. Hammedeh, S. Saidi, V. Prevelakis, and R. Ernst, “Prediction of abnormal temporal behavior in real-time systems,” in Proceedings of the 33rd Annual ACM Symposium on Applied Computing - SAC ’18. New York, New York, USA: ACM Press, 2018, pp. 359–367.

[18] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, “Contego: An Adaptive Framework for Integrating Security Tasks in Real-Time Systems,” in 29th Euromicro Conference on Real-Time Systems (ECRTS 2017), ser. Leibniz International Proceedings in Informatics (LIPIcs), M. Bertogna, Ed., vol. 76. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017, pp. 23:1–23:22.

[19] J. P. Lehoczky, “Fixed priority scheduling of periodic task sets with arbitrary deadlines.” in RTSS. IEEE Computer Society, 1990, pp. 201–209.

[20] J. Diemer, J. Rox, and R. Ernst, “Compositional Performance Analysis in Python with pyCPA,” in WATERS 2012. [Online]. Available: http://retis.sssup.it/waters2012/accepted/102_Final_paper.pdf

[21] E. Bini and G. C. Buttazzo, “Measuring the performance of schedulability tests,” Real-Time Syst., vol. 30, no. 1–2, p. 129–154, May 2005. [Online]. Available: https://doi.org/10.1007/s11241-005-0507-9

[22] “IEEE Standard for Ethernet,” IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2015), pp. 1–5600, 2018.

LEA SCHÖNBERGER received a B.Sc. degree in computer science from the University of Münster, Germany, in 2014 and a M.Sc. degree in computer science from TU Dortmund University, Germany, in 2017. Currently, she pursues a PhD in computer science. Since September 2017, she is a research associate at TU Dortmund University, where she was a member of the the Design Automation for Embedded Systems Group at the Department of Computer Science, before she switched to a half-time position in the Databases and Information Systems Group at the Department of Computer Science along with a half-time position in the AG Embedded Systems at the Department of Electrical Engineering and Information Technology in August 2020. Her research targets the field of embedded and distributed systems and focuses particularly on end-to-end quality of service guarantees and autonomous systems.

Lea Schönberger served as a reviewer for the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) and as a member of the artifact evaluation committee of the IEEE Real-Time Systems Symposium (RTSS) 2020.

MOHAMMAD HAMAD received the B.Eng. degree in software engineering and information system from Aleppo University, Syria, in 2009 and the Ph.D. (Dr.-Ing.) degree in computer engineering from the Institute for Data Technology and Communication Networks at the Technical University of Braunschweig, Germany, in 2020. Since 2020, he is a postdoctoral researcher in the Embedded Systems and Internet of Things group in the Faculty of Electrical Engineering and Information Technology at the Technical University of Munich (TUM). His research interests are in the area of Autonomous vehicle and IoT security.
JAVIER VELASQUEZ GOMEZ received the B.S. degree in electronic engineering from the National University of Colombia in 2005, the M.S. degree in business administration in technology management from the Northern Institute of Technology, Hamburg in 2019, and the M.Sc. degree in information and communication system from the Hamburg University of Technology, Germany, in 2020.

Since 2020, he is working with NXP Semiconductors where he is managing innovation projects in the field of TSN and wireless communication, especially 5G mobile networks for industrial applications.

SEBASTIAN STEINHORST received the MSc (Dipl.-Inf.) degree in computer science from the Goethe University Frankfurt, Germany, in 2005 and the Ph.D. (Dr. phil. nat.) degree in computer science from the same university in 2011.

He is an associate professor at Technical University of Munich (TUM) in Germany. He leads the Embedded Systems and Internet of Things group in the Department of Electrical and Computer Engineering. He was also a co-program PI in the Electrification Suite and Test Lab of the research center TUMCREATE in Singapore. The research of Prof. Steinhorst centers around design methodology and hardware/software architecture co-design of secure distributed embedded systems for use in IoT, automotive and smart energy applications.

SELMA SAIDI received in 2013 a Ph.D. degree in computer sciences from the University of Grenoble in France conducted together with STMicroelectronics.

After her PhD, she joined the Technical University of Braunschweig as a postdoctoral researcher and later Hamburg University of Technology as a permanent research scientist. Since 2020, she become a professor of Embedded Systems in TU Dortmund. Her research focus involve the design, implementation and validation of innovative intelligent embedded systems. Key aspects are the development of novel hardware and software design methods for embedded and autonomous systems where performance, predictability and self-adaptability play an important role. Domains of applications are avionics, autonomous driving and Internet of Things. She participates regularly as TPC member in leading conferences in embedded systems design and real-time systems such as DAC, DATE and RTSS.