Securing the Precision Time Protocol with SDN-enabled Cyclic Path Asymmetry Analysis

ANDREAS FINKENZELLER, Technical University of Munich, Germany
ARNE FUCKS, Technical University of Munich, Germany
EMANUEL REGNATH, Siemens AG, Germany
MOHAMMAD HAMAD, Technical University of Munich, Germany
SEBASTIAN STEINHORST, Technical University of Munich, Germany

High-precision time synchronization is a vital prerequisite for many modern applications and technologies, including Smart Grids, Time-Sensitive Networking (TSN), and 5G networks. Although the Precision Time Protocol (PTP) can accomplish this requirement in trusted environments, it becomes unreliable in the presence of specific cyber attacks. Mainly, time delay attacks pose the highest threat to the protocol, enabling attackers to diverge targeted clocks undetected. With the increasing danger of cyber attacks, especially against critical infrastructure, there is a great demand for effective countermeasures to secure both time synchronization and the applications that depend on it. However, current solutions are not sufficiently capable of mitigating sophisticated delay attacks. For example, they lack proper integration into the PTP protocol, scalability, or sound evaluation with the required microsecond-level accuracy. This work proposes an approach to detect and counteract delay attacks against PTP, which is based on cyclic path asymmetry measurements over redundant paths. We leverage Software Defined Networking (SDN) capabilities to dynamically find these redundant paths in arbitrary networks, recommend new links to increase the network's security, and ensure deterministic routing. Furthermore, we show how path redundancy can be utilized to reveal and mitigate undesirable asymmetries on the synchronization path that cause the malicious clock divergence. Moreover, we propose PTPsec, a secure PTP protocol, and its implementation based on the latest IEEE 1588-2019 standard. With PTPsec, we advance the conventional PTP to support reliable delay attack detection and mitigation. We validate our approach in software simulations and on a hardware testbed, which includes an attacker capable of performing static and incremental delay attacks at a microsecond precision. Our experimental results show that the proposed approach is scalable, and all attack scenarios can be reliably detected and mitigated with minimal detection time.

CCS Concepts: • **Security and privacy → Security protocols**; • **Networks → Network protocol design**.

Additional Key Words and Phrases: Security, IEEE 1588, PTP, Time Delay Attack, Time Synchronization, SDN

ACM Reference Format:
Andreas Finkenzeller, Arne Fucks, Emanuel Regnath, Mohammad Hamad, and Sebastian Steinhorst. 2025. Securing the Precision Time Protocol with SDN-enabled Cyclic Path Asymmetry Analysis. *ACM Trans. Cyber-Phys. Syst.* 1, 1 (March 2025), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' Contact Information: Andreas Finkenzeller, andreas.finkenzeller@tum.de, Technical University of Munich, Munich, Germany; Arne Fucks, arne.fucks@tum.de, Technical University of Munich, Munich, Germany; Emanuel Regnath, emanuel.regnath@siemens.com, Siemens AG, Munich, Germany; Mohammad Hamad, mohammad.hamad@tum.de, Technical University of Munich, Munich, Germany; Sebastian Steinhorst, sebastian.steinhorst@tum.de, Technical University of Munich, Munich, Germany.

ACM 2378-9638/2025/3-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Precise time synchronization is indispensable for many applications and technologies, such as Smart Grids, Time Sensitive Networking (TSN), and 5G networks. Especially TSN facilitates various time-critical use cases across different domains, including aerospace, automotive, Audio Video Bridging (AVB), and industrial automation. To function properly, these applications typically require synchronization accuracies at a microsecond level, and even minor deviations can already have significant impacts. The consequences range from performance degradation to, in the worst case, complete system failure [33]. The 802.1AS synchronization standard [18] on which TSN is based on, for example, guarantees a peak-to-peak synchronization in steady-state of 1 µs. Synchrophasor measurements and Phasor Measurement Units (PMUs) enable data-driven applications in Smart Grids and demand maximum clock synchronization errors of approx. 20 µs to 30 µs for reliable phase angle measurements [16, 25].

The Precision Time Protocol (PTP) (§ 2.1) is a commonly used network protocol to provide accurate time synchronization in the scenarios mentioned above. However, PTP was initially designed without any security considerations in mind and, hence, makes certain network assumptions, such as path symmetry and message integrity, that might not always hold. Especially in spatially large networks like Smart Grids, cyber attacks targeting the network infrastructure are becoming more likely, forcing the protocol designers to reconsider security concepts in PTP. With the four-pronged approach outlined in Annex P of the latest 2019 revision [15], the PTP standard aims to address this problem. While the suggested cryptographic countermeasures can guarantee message integrity and protect against many other attack vectors, including message replay and spoofing attacks, they cannot counteract all known threats. In particular, time delay attacks (§ 2.2) remain an unsolved problem, as previous work has shown extensively [4, 5, 9].

Delay attacks violate the protocol's assumption of symmetric path delays by maliciously introducing unidirectional delays into the network. In the literature, initial approaches have recently been proposed to counteract delay attacks against PTP, none of which provide an effective solution to the problem yet. Threshold-based approaches, for example, as presented in [22, 36], perform the attack detection by continuously monitoring the reported path delay and comparing it to a previously defined threshold. However, the threshold definition is quite challenging. Static thresholds must be set conservatively to avoid false alarms, while dynamic thresholds, which adapt based on previous measurements, cannot adequately protect the system from incremental delay attacks. Moreover, the reported path delay is not guaranteed to change at all during a successful delay attack [32]. Other solutions, including additional network guards [1, 30] or special monitoring and reporting mechanisms [28, 31] only work within a limited attacker model. Because the detection method is not sufficiently coupled to the time synchronization mechanism, attackers can distinguish PTP messages from other network traffic and handle them differently to bypass existing mitigations. Besides, there are first efforts to countermeasures based on path redundancy. However, the available works [26, 32] only present some general ideas that, among other things, lack scalability and proper protocol integration to become applicable. Hence, there is still a great need for a secure and comprehensive solution that entirely protects PTP from time delay attacks.

Contributions: This work investigates cyclic Round-Trip Time (RTT) measurements to analyze network path asymmetries and their applicability for delay attack detection and mitigation in time synchronization protocols. The cyclic analysis is enabled by redundant network paths, which play a decisive role here. Each PTP event message invokes a subsequent measurement packet that is returned to the originator via a previously determined redundant path to complement the cyclic RTT measurement. The measurement packets are dedicated messages we introduce in our proposed PTPsec protocol to entangle the attack detection with the default synchronization procedure in

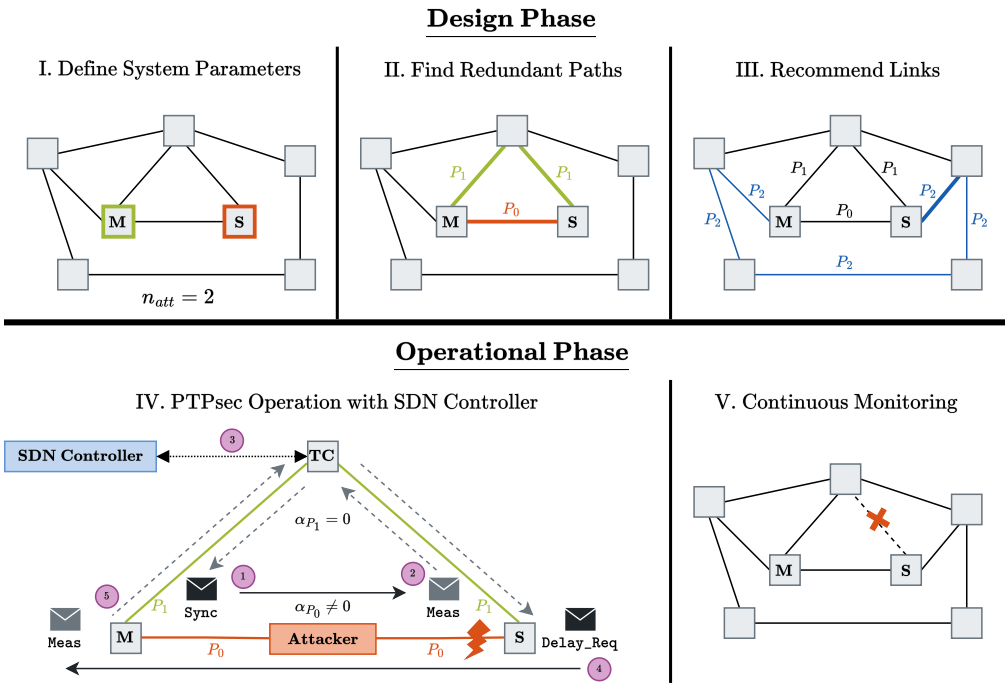


Fig. 1. The proposed workflow for secure time synchronization with PTPsec. After setting the system parameters (I.), the given network is analyzed to find existing redundant paths (II.) and recommend additional links if needed (III.) to meet certain security requirements. In the operational phase, we apply our path asymmetry analysis concept to detect time delay attacks within the proposed PTPsec protocol (IV.): The PTP Sync message, sent via the attacked path P_0 (1), is followed by our newly introduced Meas message over the genuine path P_1 to complete the first round trip (2). The SDN controller ensures correct packet routing (3). Similarly, the exchange of the Delay_Req (4) and another Meas message (5) leads to a second circulation. This allows for cyclic RTT measurements from which we derive the current path asymmetry α_{P_0} to reveal and mitigate ongoing delay attacks. We monitor the network live, enabling quick responses to dynamic system changes (V.).

conventional PTP. This prevents any circumvention of our introduced protection mechanism. For the final detection and mitigation of ongoing delay attacks, we analyze the path asymmetry based on the two RTT measurements obtained in one PTP synchronization round, postulate ongoing attacks if non-zero, and compensate for the asymmetry[1] accordingly.

Furthermore, we present a Software-defined Networking (SDN)-based approach to implement our proposed concepts for secure time synchronization. Fig. 1 illustrates the entire implementation process consisting of the design and operational phases. First, we model the system and set all relevant system parameters, including graph modeling and defining the required security demands (I., § 2). Second, we analyze the given network and find all existing redundant paths (II., § 6.3). If there are too few, we recommend new links that can be added to meet the path redundancy requirements (III., § 6.4). The operational phase mainly consists of running our PTPsec protocol (IV., § 5). To ensure correct routing behavior and perform continuous monitoring, we leverage SDN capabilities for dynamic network management (V.,§ 6.5). To the best of our knowledge, the

[1]This also includes any type of non-malicious link asymmetries in the network.

presented procedure is the first comprehensive approach that efficiently detects and mitigates time delay attacks and, consequently, performs high-precision time synchronization securely. In conclusion, in this work, we:

- analyze cyclic RTT measurements and show their effectiveness for reliable path asymmetry detection (§ 3),
- derive a theoretical model for time delay attack detection and mitigation in arbitrary networks (§ 4),
- present PTPsec, a secure PTP protocol including its implementation as an extension of the latest IEEE 1588-2019 standard that adopts our proposed attack countermeasures (§ 5),
- introduce an SDN approach for dynamic network management to find redundant paths and ensure proper packet routing (§ 6), and
- extensively evaluate our PTPsec implementation both in software simulations and on a realistic hardware testbed to confirm that our approach successfully detects and mitigates time delay attacks (§ 7).

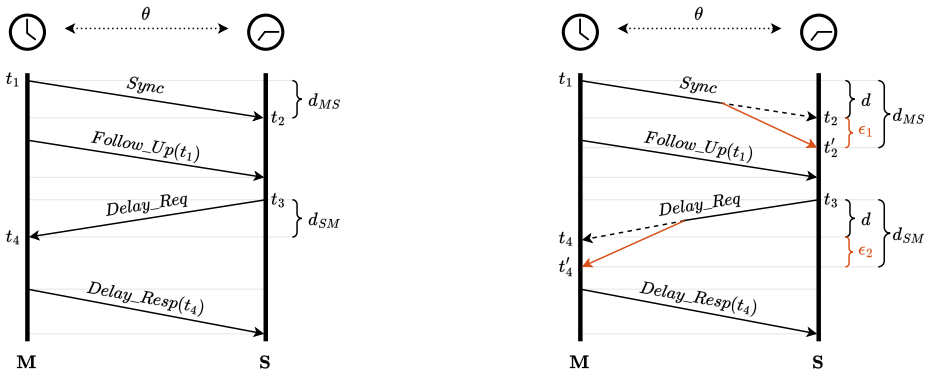
Note that parts of this work have already been published in a preceding conference paper at IEEE Infocom 2024 [8]. In this article, we advance the initial ideas and incorporate them into a complete approach to secure high-precision time synchronization with PTP. Specifically, we introduce SDN as the enabling technology to run required network tasks, such as dynamic topology discovery, redundant path management, and deterministic routing. We present a new algorithm recommending links to increase path redundancy and discuss all other SDN-related aspects in more detail. Furthermore, we present new hardware measurements with improved performance and also include additional simulation results to show PTPsec's scalability. Last but not least, we also update our related works section to include the most recent work and extend the comparison regarding applicability, extent of experiments, and achieved detection accuracy.

2 System Model

2.1 Precision Time Protocol

The precision time protocol, as defined in the IEEE 1588 standard [15], is a network protocol that synchronizes devices with an accuracy of less than 1 µs. The standard considers various clock types to account for different device behaviors in typical PTP networks. Particularly, Ordinary Clocks (OCs) denote master and slave nodes[2] which either provide a time reference to others or synchronize their clocks to a specific master, respectively. In contrast, Transparent Clocks (TCs) do not actively synchronize to other clocks but are transparent switching devices that connect multiple OCs to facilitate extensive networks. The synchronization procedure comprises two phases. In the protocol's first phase, all participating nodes announce their existence and determine the best (i.e., most accurate) clock in the network, denoted the *Grandmaster* clock, using the Best Master Clock Algorithm (BMCA). In the second phase, all other nodes synchronize their clocks to the *Grandmaster* by periodically exchanging PTP messages. Along with the exchange of `Sync` and `Delay_Req` messages, four timestamps are captured at both the transmission and reception of each packet in the End-to-End (E2E) mode. Here, using hardware timestamping significantly increases the synchronization accuracy. The two so-called event messages are critical for the achieved synchronization accuracy since the obtained timestamps are the basis for subsequent offset calculations. With the additional assumption of symmetric path delays ($d_{MS} = d_{SM}$), which is an indispensable assumption for all network-based time synchronization protocols, including PTP, the current clock offset θ can be computed as:

[2]In this work, we stick to the terminology used in the official PTP standard document to avoid any confusion besides being potentially inappropriate.



(a) PTP (b) Delay Attack

Fig. 2. a) PTP message flow with timestamping to minimize the clock offset θ. b) When attackers can delay PTP event messages (Sync or Delay_Req), they create path asymmetries that impair the clock synchronization.

$$\theta = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{1}$$

where t_1, t_2, t_3, and t_4 denote the captured timestamps of the two event messages. The diagram in Fig. 2a depicts the taken timestamps along the entire message flow of one synchronization round. Note that the Follow_Up message conveying the captured transmission timestamp t_1 is only required when PTP operates in two-step mode.

2.2 Time Delay Attack

However, if the synchronization path is not symmetric due to malicious network activities, the offset calculations are based on an invalid assumption, leading to erroneous synchronization results. So, if attackers were able to deliberately delay the Sync and Delay_Req messages by ϵ_1 and ϵ_2 respectively with $\epsilon_1 \neq \epsilon_2$, as illustrated in Fig. 2b, the clock offset θ' would mistakenly compute as:

$$\begin{aligned} \theta' &= \frac{((t_2 + \epsilon_1) - t_1) - ((t_4 + \epsilon_2) - t_3)}{2} \\ &= \frac{(t_2 - t_1) - (t_4 - t_3)}{2} + \frac{\epsilon_1 - \epsilon_2}{2} \\ &= \theta + \frac{\alpha}{2} \end{aligned} \tag{2}$$

with $\alpha = d_{MS} - d_{SM} = \epsilon_1 - \epsilon_2$. Provided $\alpha = 0$, the path is perfectly symmetric, and the time synchronization works as intended. However, if the delay attack is successful and $\alpha \neq 0$, the path becomes asymmetric, and the synchronized clock runs behind or ahead in time of the *Grandmaster* clock, respectively. The described time delay attack is not only theoretical in nature but can also be practically implemented in various ways. For example, Barreto et al. [5] present a static delay attack scenario, where the attackers deliberately extend the optical fibers in Smart Grid networks to add a constant one-way delay, turning the synchronization path asymmetric. In [9], the authors perform an incremental delay attack, where the introduced asymmetric path delay gradually increases, to demonstrate more dynamic attacks. Both works show the general feasibility of time delay attacks and emphasize their devastating impact on PTP, highlighting the urgent need for sound countermeasures.

2.3 Network Model

A typical PTP deployment comprises an entire network, including many devices that shall be synchronized. Therefore, we model a given PTP network as the graph $\mathcal{G} = (V, E)$ with $V = \{v_0, \ldots, v_n\}$, $n \in \mathbb{N}$ denoting the set of vertices and $E = \{e_0, \ldots, e_m\} \subseteq V \times V$, $m \in \mathbb{N}$ the set of edges. Each vertex $v_i \in V$ represents a node participating in the PTP protocol. For proper synchronization, a minimal PTP network requires at least the elected *Grandmaster* node and one additional slave device that synchronizes to the master. Thus, we assume our vertex set V to contain these two nodes at the minimum, which we denote for further reference as $M \in V$ for the master and $S \in V$ for the slave node, respectively. All edges $e_i \in E$ are considered bidirectional and model the connecting links between all involved network devices. Furthermore, we assume $\mathcal{G}$ to be connected. Hence, at least one path connects M and S for the PTP message exchange, which we denote as the synchronization path P_0. Also, we extend the notation of α, which we introduced in § 2.2 to express the asymmetry α_{e_i} of a specific edge $e_i \in E$. Additionally, we define the path asymmetry α_{P_i}, which is experienced by all traversing packets on path P_i, as the sum of the individual link asymmetries of all composing links:

$$\alpha_{P_i} = \sum_j \alpha_{e_j}, \ \forall e_j \in P_i. \tag{3}$$

Furthermore, we assume full network control, including knowledge about the entire topology and the ability to set packet routes. For that, we leverage SDN capabilities with a centralized SDN controller managing all network tasks as further explained in § 6. We assume that all participating nodes follow SDN-related communication and tasks honestly.

2.4 Attacker Model

Many attacks exist against time synchronization protocols, particularly against PTP [3]. In this work, however, we mainly focus on time delay attacks and related attacker capabilities due to their criticality and define our attacker model accordingly. Similar to [5, 9], we deny the attackers internal access to the protocol, i.e., the possibility to directly modify PTP header fields. If this was possible, the attackers were powerful enough to manipulate the synchronization on a protocol level by changing timestamps and the correction field contents, which would render delay attacks superfluous. Security protocols, such as IPsec and MACsec, can be appropriate remedies to ensure the demanded message integrity. Furthermore, attackers are not allowed to compromise hosts or intermediary network devices in a way that would grant them internal protocol access or allow them to update the clocks directly with a similar reasoning.

Nevertheless, we assume the attackers to have full knowledge of the network topology, including knowledge about deployed PTP devices and clock types. Additionally, they are fully aware of any implemented protection methods (e.g., IPsec). After compromising a link, the attackers can delay all passing packets individually. More precisely, they can deliberately and independently add unidirectional delays to selected messages in both transmission directions. This fine-grained packet delay is even possible with specific protection mechanisms enabled, such as traffic encryption, as shown in [4]. Moreover, the attackers can simultaneously perform multiple delay attacks precisely synchronized at different locations in the network if they compromise more than one link. There are no restrictions on the number of compromised links or the strategy of how attackers perform joint attacks. However, we assume that the selection of hijacked links will remain constant for a sufficiently long period to allow for steady-state analysis. Particularly, the attackers may freely change the introduced delay for an attacked link over time but not attack a different link. Finally, we require all nodes participating in the PTP protocol to be honest, i.e., to follow the protocol as specified in the standard or our proposed PTPsec protocol.

3 Asymmetry Analysis

If we could precisely measure unidirectional delays in the network, time delay attacks would be easily detected. However, accurately measuring one-way path delays is a challenging problem in network theory [7]. In the following analysis, we benefit from the work of Gurewitz et al. [13, 14] on one-way delay estimation, which we extend to derive our cyclic path asymmetry analysis. Additionally, we propose a general path asymmetry model that forms the basis for the delay attack detection and mitigation methods presented in § 4. First, we start with a simple two-node example from which we develop the general model for arbitrary networks.

3.1 Two-Node Scenario

Given a simple synchronization scenario with only two nodes M and S and one attacked edge e_0, as shown in Fig. 3a, we are trying to estimate the link asymmetry $\alpha_{e_0} \neq 0$. However, it is impossible to reliably determine α_{e_0} with only one available link. Thus, we need at least one additional link to obtain a circular structure that we can use for further analysis.

3.1.1 One redundant link. If there exists a second edge e_1 which is redundant to e_0, both edges form a cycle, as depicted in Fig. 3b. Now, we can perform a cyclic RTT measurement using e_0 and e_1 by forwarding a packet from M to S on link e_0 which is subsequently returned to M via e_1. Upon arrival at node M, the elapsed time RTT_{e_0,e_1} computes as:

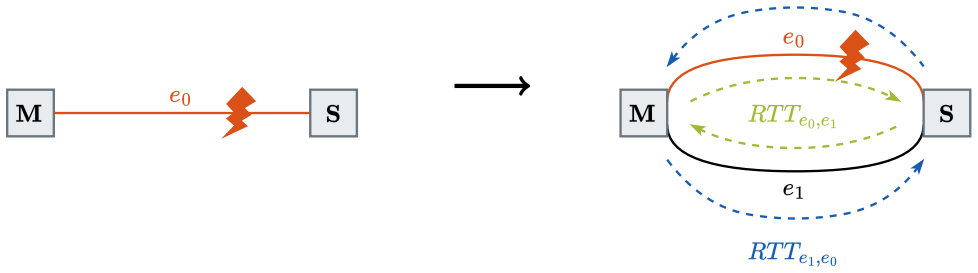
$$RTT_{e_0,e_1} = t_{in} - t_{eg} \tag{4}$$

where t_{in} is the packet's measured ingress and t_{eg} the egress timestamp at node M. Additionally, we conduct a second RTT measurement RTT_{e_1,e_0} with reverse transmission direction, i.e., forwarding the packet on link e_1 to S and returning it via e_0. Both measurements include unidirectional delays of e_0 but in opposite directions. For RTT_{e_0,e_1}, edge e_0 contributes in forward direction (M to S) while RTT_{e_1,e_0} contains its delay in backward direction (S to M). If we further assume that link e_1 is perfectly symmetric ($\alpha_{e_1} = 0$), i.e., the contributed link delay is equal in both directions, we can calculate the asymmetry of link e_0 as:

$$\alpha_{e_0} = RTT_{e_0,e_1} - RTT_{e_1,e_0} \tag{5}$$

Note that the opposing one-way delays of link e_1 cancel out due to its symmetry, and we are left with the desired difference of both unidirectional delays of e_0. Here, the essential part is the cyclic measurement approach capturing the opposing one-way delays of a single link isolated in distinct measurements. Furthermore, the two RTTs measurements originate from the same clock, avoiding the need for any time synchronization. Also, note that the exact measurements could have likewise been taken on node S since the starting point in a cyclic measurement is irrelevant.

3.1.2 N redundant links. The asymmetry measurement only worked in the previous example with one redundant edge because we assumed the additional link e_1 to be symmetric. However, if the second link was also asymmetric due to another delay attack, for example, attackers could thwart the measurement attempt while still successfully introducing unidirectional delays on e_0. For that, they need to add similar one-way delays ϵ to both links e_0 and e_1 but in opposing directions so that the delays cancel out each other in (5) while the individual links become asymmetric. In such a case, we need another edge e_2 with symmetric link delay for compensation. This additional link increases the number of cycles in our network, allowing for further independent cyclic measurements. With two redundant links, we can perform two asymmetry measurements that include e_0, leading to the following equations:



(a) One Attacked Link (b) Two Links for Cyclic Structure

Fig. 3. Cyclic path asymmetry analysis illustrated with two nodes. While there is no efficient method to calculate the asymmetry with only one link (a), a second link enables a cyclic structure for further analysis (b). Two RTT measurements in opposing directions can be smartly combined to determine the link asymmetry α_{e_0} of the attacked link e_0.

$$\begin{aligned} \alpha_{e_0}^{(1)} &= RTT_{e_0,e_1} - RTT_{e_1,e_0} \\ \alpha_{e_0}^{(2)} &= RTT_{e_0,e_2} - RTT_{e_2,e_0} \end{aligned} \tag{6}$$

with $\alpha_{e_0}^{(1)}$ and $\alpha_{e_0}^{(2)}$ being independent estimates for the targeted link asymmetry α_{e_0}. Although $\alpha_{e_0}^{(1)}$ might equal to zero, indicating perfect link symmetry due to the mentioned attack strategy, $\alpha_{e_0}^{(2)}$ yields the correct estimate because of the demanded link symmetry of e_2 and the reasoning of (5).

Similarly, we can derive the general case of n redundant edges (in addition to e_0) for the two-node scenario. From $2n$ pairwise RTT measurements RTT_{e_0,e_i} and RTT_{e_i,e_0}, $i \in [1, n]$, we get n estimates for the link asymmetry α_{e_0}:

$$\begin{aligned} \alpha_{e_0}^{(1)} &= RTT_{e_0,e_1} - RTT_{e_1,e_0} \\ \alpha_{e_0}^{(2)} &= RTT_{e_0,e_2} - RTT_{e_2,e_0} \\ &\vdots \\ \alpha_{e_0}^{(n)} &= RTT_{e_0,e_n} - RTT_{e_n,e_0} \end{aligned} \tag{7}$$

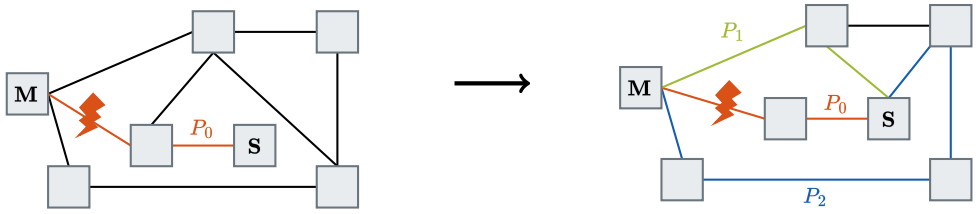
Furthermore, we can state the following important finding:

> **Finding 1:** In order to successfully determine α_{e_0}, we need at least one link e_i, $i \in [1, n]$ with symmetric link delay $\alpha_{e_i} = 0$, which is redundant to e_0. Using this symmetric link e_i, the asymmetry measurement yields a correct estimate $\alpha_{e_0}^{(i)} = RTT_{e_0,e_i} - RTT_{e_i,e_0} = \alpha_{e_0}$.

This necessary condition directly results from the previous reasoning that attackers could introduce various unidirectional delays to cancel out each other in the cyclic measurements if we allowed all links to be asymmetric.

3.2 Multi-Node Scenario

Realistic networks usually comprise more than two nodes. Therefore, we need to derive a general path asymmetry model that applies to networks of any size. Similar to the two-node scenario, we



(a) One Attacked Path (b) Multiple Redundant Paths

Fig. 4. Redundant path principle in multi-node networks. To efficiently estimate the path asymmetry α_{P_0}, we require other edge-disjoint paths P_i to enable cyclic RTT measurements.

attempt to estimate the asymmetry α_{P_0} on the synchronization path P_0. We equally require at least one redundant path P_1 in addition to P_0, as exemplified in Fig. 4, to enable the cyclic measurements previously proposed in § 3.1. Note that P_1 must not share any edge with P_0 to be considered fully redundant since all edges of P_0 are contributing to its total path asymmetry as defined in (3). If a single joint edge existed, it would affect both paths equally and make a cyclic measurement with independent forward and backward delays impossible. Therefore, we need P_0 and P_1 to be edge-disjoint:

$$P_0 \cap P_1 = \emptyset \tag{8}$$

With this requirement, we can formulate the general approach to estimate path asymmetries in arbitrary networks. Given the network $\mathcal{G} = (E, V)$, the synchronization path P_0, and n redundant paths $P_i,\ i \in [1, n]$ connecting nodes M and S, we perform $2n$ cyclic RTT measurements for each pair P_0 and $P_i,\ i \in [1, n]$. From these measurements, we derive the following set of equations to get n estimates for the path asymmetry α_{P_0}:

$$\begin{aligned} \alpha_{P_0}^{(1)} &= RTT_{P_0,P_1} - RTT_{P_1,P_0} \\ \alpha_{P_0}^{(2)} &= RTT_{P_0,P_2} - RTT_{P_2,P_0} \\ &\vdots \\ \alpha_{P_0}^{(n)} &= RTT_{P_0,P_n} - RTT_{P_n,P_0} \end{aligned} \tag{9}$$

Moreover, we state the generalized finding from our path asymmetry analysis in arbitrary networks:

> **Finding 2:** For all symmetric paths $P_i,\ i \in [1, n]$ with $\alpha_{P_i} = 0$, we get correct estimates $\alpha_{P_0}^{(i)} = RTT_{P_0,P_i} - RTT_{P_i,P_0} = \alpha_{P_0}$. Hence, we require at least one redundant path to be symmetric to successfully determine the desired path asymmetry α_{P_0} for the synchronization path P_0.

4 Attack Detection and Mitigation

With this general path asymmetry model, we can now define the detection criteria for the delay attack and derive effective mitigation techniques.

4.1 Detection Criteria

From (2), we know that successful time delay attacks result in non-zero path asymmetries. Hence, we can use the derived analysis from § 3 to find asymmetric paths and, consequently, detect ongoing

attacks in PTP networks. With $n+1$ total edge-disjoint paths $P_i,\ i \in [0, n]$ connecting nodes M and S, where P_0 is used for PTP synchronization, we yield n independent asymmetry estimates for α_{P_0}, as stated in (9). Based on these estimates, we define the detection criterion for an ongoing delay attack as follows:

$$delay\_attack \leftarrow \begin{cases} True, & \exists\, \alpha_{P_0}^{(i)} \neq 0,\ i \in [1, n] \\ False, & \text{otherwise} \end{cases} \tag{10}$$

Additionally, we know that the cyclic measurements only work if at least one path is symmetric. Thus, we can successfully detect delay attacks that include up to n attacked paths. Note that the selection of P_0 can be arbitrary without violating the criterion stated in (10). However, since P_0 can be selected randomly, the attack's actual impact on the synchronization depends on the specific scenario. We consider the following possibilities in the worst-case scenario when n paths are attacked[3]: (1) Assuming the selected synchronization path P_0 is attacked, then the measurement resulting from the one demanded benign path P_j reveals this asymmetry, i.e., $\alpha^{(j)} \neq 0$, and the detection criterion is correct. (2) If P_0 denotes the genuine path, all non-zero estimates indicate an attack on the respective redundant paths, while the actual synchronization path is not affected. Hence, using (10) in this scenario leads to false positives as an attack would be detected without any impact on the actual synchronization. Consequently, we need to localize the attack more precisely in the network to gain sufficient certainty about its effective impact on the synchronization.

For that, we first analyze all estimated path asymmetries $\alpha^{(i)},\ i \in [1, n]$ and search for potential attack scenarios that match the obtained result. Particularly, we cluster all paths into two groups, where one group contains all genuine and the other all attacked paths. However, the clustering only works partially. While the general segmentation into two classes is possible, the final class labeling cannot be made without further assumptions. In the scenario above, this results in either P_0 being benign and the other paths $P_i\ i \in [1, n]$ attacked or vice versa. Both configurations lead to similar estimations $\alpha_{P_0}^{(i)} \neq 0,\ \forall i \in [1, n]$. Hence, we limit the number of attackers to an upper bound of $n_{att} \leq \lfloor \frac{n}{2} \rfloor$ to further distinct the two groups. With this additional assumption, we can state a more targeted criterion for attack detection in PTP networks:

$$(delay\_attack, P_0\ affected) \leftarrow \begin{cases} (True, True), & \exists\, \alpha^{(i)} \neq 0,\ i \in [1, n] \wedge n_{att} \leq \lfloor \frac{n}{2} \rfloor \\ (True, Maybe), & \exists\, \alpha^{(i)} \neq 0,\ i \in [1, n] \wedge n_{att} > \lfloor \frac{n}{2} \rfloor \\ (False, False), & \text{otherwise} \end{cases} \tag{11}$$

So, we conclude that with n redundant paths, we can detect up to n attacked links in the network. Moreover, we can localize the attacks and determine the impact on the synchronization path P_0 if we bound the number of attacks to $n_{att} \leq \lfloor \frac{n}{2} \rfloor$.

4.2 Attack Mitigation

If an attack is detected, we can use the measured path asymmetry α_{P_0} of the synchronization path P_0 to mitigate the ongoing attack. For that, we first need to derive α_{P_0} from our obtained asymmetry estimates in (9). In this matter, we can follow different strategies. If there is only one estimate ($n = 1$), it is naturally our best candidate for the actual path asymmetry:

$$\alpha_{P_0} = \alpha_{P_0}^{(1)} \tag{12}$$

[3]The reasoning for scenarios with fewer attackers works analogously.

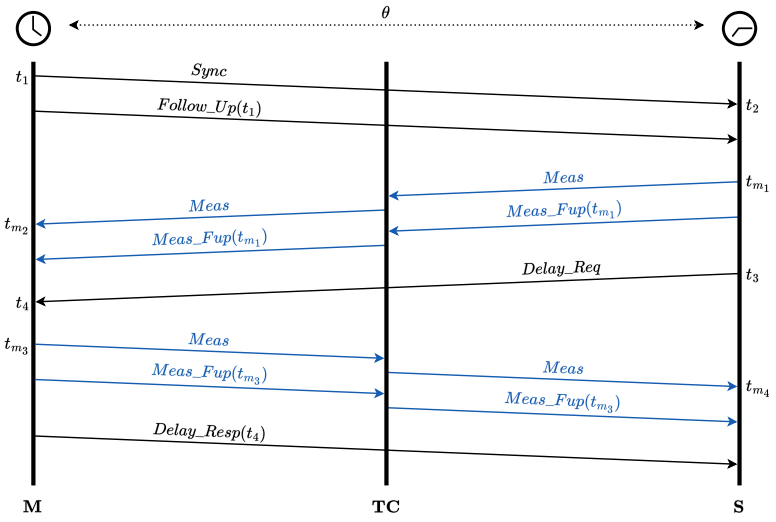


Fig. 5. Proposed PTPsec message flow to protect against time delay attacks. After the reception of PTP event messages (Sync and Delay_Req), dedicated (Meas) messages are returned to the originator via a redundant network path, here indicated with the additional TC, to enable cyclic path asymmetry measurements. The Meas_Fup messages convey the captured timestamps t_{m_1} and t_{m_3}, respectively, when PTP operates in two-step mode.

In case there exist various redundant paths ($n \geq 2$), and no further information about link quality or the attack strategy is available, all estimates can be considered equally suitable. Consequently, α_{P_0} results in the mean of all non-zero measurements:

$$\alpha_{P_0} = \frac{1}{|I|} \sum_{i \in I} \alpha_{P_0}^{(i)}, \; I = \{i \mid \alpha_{P_0}^{(i)} \neq 0 \wedge 1 \leq i \leq n, \; i \in \mathbb{N}\} \tag{13}$$

Alternatively, using the median value instead could be a more robust option as it is less affected by outliers. With α_{P_0}, we can further calculate the rectified clock offset θ_{rect} that is compensating malicious path asymmetries using the reported PTP offset θ_{rep} from (1) and the insight gained from (2):

$$\theta_{rect} = \theta_{rep} - \frac{\alpha_{P_0}}{2} \tag{14}$$

This rectified clock offset can be used as input for PTP's control algorithm to securely update the local clock oscillator despite any ongoing attack.

5 Proposed PTPsec

As already highlighted in § 2.1, the latest IEEE 1588-2019 standard (PTP v2.1) is not secure against time delay attacks. Hence, we propose the PTPsec protocol, which integrates appropriate countermeasures into the existing IEEE standard to improve PTP's resilience against such attacks. Recall that in our attacker model, attackers can precisely delay individual packets. Particularly, they could selectively delay only PTP messages if these were distinguishable and independent from our measurement packets. Therefore, we need to entangle the RTT measurements with the PTP

synchronization procedure to prevent bypassing the presented attack detection and mitigation approach. For that, we aim to integrate the two critical event messages `Sync` and `Delay_Req` directly into the RTT measurements. Consequently, we ensure that deliberate packet delays, which impede the time synchronization, affect the proposed detection approach equally.

Let RTT_{P_0,P_i} be the RTT measurement including the synchronization path P_0 and a redundant path P_i to estimate the path asymmetry α_{P_0}. Then, the `Sync` message, sent from node M to S via P_0, already initiates the first RTT measurement. Once received at node S, an immediate response is returned to M via the redundant path P_i to finish the round trip. Since this response has a different purpose than existing PTP message types, we introduce the new message type `Meas` in our proposed PTPsec protocol. Similar to `Sync` messages, also `Meas` packets are PTP event messages due to the requested timestamps at packet transmission and reception. We denote the captured `Meas` timestamps with an additional m, e.g., t_{m_1}, to distinguish them from the four default PTP timestamps. If the protocol operates in two-step mode, we require an additional new message `Meas_Fup` to forward the captured timestamp in a separate follow-up message. To continue the message flow, S sends the expected `Delay_Req` message via P_0 to M which is immediately followed by another `Meas` message to S via P_i upon reception to finish the second measurement cycle yielding RTT_{P_i,P_0}. Finally, M answers the request with a `Delay_Resp` message to complete the synchronization protocol. Fig. 5 shows the full sequence of our proposed PTPsec protocol. With all captured timestamps in this iteration, the two RTT measurements compute as:

$$\begin{aligned} RTT_{P_0,P_i} &= (t_{m_2} - t_1) - (t_{m_1} - t_2) \\ RTT_{P_i,P_0} &= (t_{m_4} - t_3) - (t_{m_3} - t_4) \end{aligned} \tag{15}$$

from which we derive the path asymmetry estimate for P_0:

$$\alpha_{P_0}^{(i)} = RTT_{P_0,P_i} - RTT_{P_i,P_0} \tag{16}$$

Note that in the presented procedure, both critical PTP event messages are fully integrated into the path asymmetry analysis. Consequently, any malicious delay equally impacts the synchronization and the employed countermeasures, preventing attackers from bypassing our approach even if individual PTP messages can be distinguished and delayed.

Since the proposed PTPsec protocol requires additional measurement packets for the asymmetry analysis, it also introduces a message overhead compared to conventional PTP. In particular, four supplementary packets are sent in two-step mode on each of the n redundant network path per PTP synchronization cycle, resulting in the following total number of PTPsec messages per cycle:

$$\#\,packets = 4 + 4n \tag{17}$$

Note the packet count's linear increase with the number of redundant paths. However, the actual number of redundant paths to consider for securing the protocol is a security parameter that can be adjusted depending on the assumed attack model. Furthermore, PTP messages usually account only for a small part of the entire network traffic, alleviating the increased packet count's impact.

6 Network Management

To find the required redundant paths and to ensure correct routing behavior for the cyclic measurements, we leverage SDN capabilities. Particularly, we employ an SDN controller to centralize the network logic. This enables the core requirements for our path asymmetry analysis and also simplifies general network management (e.g., dynamic topology changes, protocol updates, etc.). Note that while SDN has many benefits, it is not a prerequisite to run PTPsec.

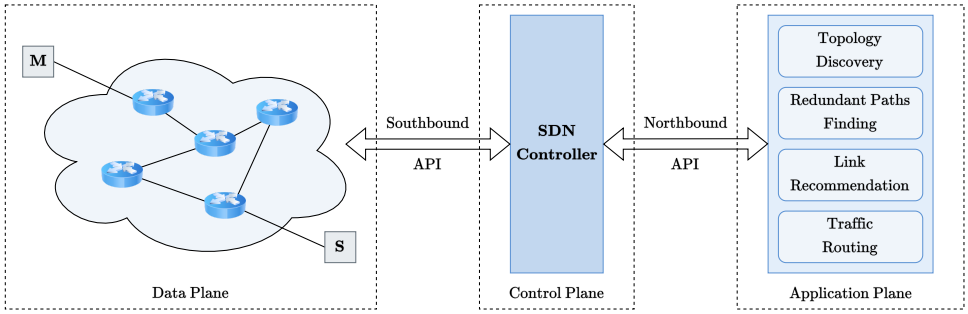


Fig. 6. Typical SDN architecture tailored to our PTPsec implementation for secure time synchronization. The desired network tasks, such as traffic routing and redundant pathfinding, run on top of the SDN controller, which is the connecting link to the data plane.

6.1 SDN-based Network Architecture

The SDN paradigm dictates a logical separation of data, control, and application planes [19] with dedicated communication interfaces denoted as the southbound and northbound Application Programming Interfaces (APIs). Fig. 6 illustrates the common SDN architecture tailored to our application. All switching devices (i.e., TCs) in the data plane do not contain any logic by default and communicate with the SDN controller, for example, via the OpenFlow protocol [24], to get appropriate directives. Whenever a new packet arrives, the switch forwards it to the SDN controller and asks for further instructions. Based on the desired network behavior, the controller determines a suitable action and commands the switch how to handle the packet. Typical actions include – but are not limited to – flooding, forwarding, and dropping incoming packets. Furthermore, the SDN controller can install specific rules in the switch's flow table to avoid the communication overhead for subsequent packets in case the desired actions are already known. The switch then checks whether any filter rule in its flow table matches the incoming packet. If so, it performs the intended action for the first match. If there exists no appropriate rule in the table, the switch forwards the packet to the controller and waits for assistance as usual. The flow table optimization is particularly helpful in saving time and network resources. Note that the overhead caused by the communication with the SDN controller does not affect the synchronization if hardware timestamping is used. The extra delay only affects the residence time within the TC, which is compensated in the synchronization process. However, if software timestamping is used instead, it will significantly degrade the accuracy of the synchronization. We define the desired network behavior and resulting actions with appropriate applications that run on top of the controller as visualized in Fig. 6. In the following, we explain the relevant tasks in more detail.

6.2 Topology Discovery

The SDN controller is a centralized logic unit that can control every packet flow in the network. This also allows for dynamic link discovery and the creation of an internal topology representation when the network operation starts up. For this, the controller instructs all switches to send Link Layer Discovery Protocol (LLDP) [17] messages and observes the resulting packet flows to construct the network graph $\mathcal{G}$ as introduced in § 2.3. This graph is the foundation for all other network functions for which precise knowledge of the current topology is necessary, such as searching redundant paths (§ 6.3).

Algorithm 1 find_edge_disjoint_paths (Adapted Ford-Fulkerson Algorithm)

Input: Network $\mathcal{G} = (V, E)$, Source $M \in V$, Sink $S \in V$
Output: Set of edge-disjoint paths $\mathcal{P}$ from M to S

1: $\mathcal{P} \leftarrow \emptyset$
2: $flow(e) \leftarrow 0$ **for each** $e \in E$
3: **while** $p \leftarrow$ find M-S path with $flow(e) = 0 \ \ \forall e \in p$ **do**
4: $\quad \mathcal{P}$.insert(p)
5: $\quad flow(e) \leftarrow 1$ **for each** $e \in p$
6: **return** $\mathcal{P}$

6.3 Redundant Paths Finding

An essential aspect of the asymmetry detection method presented in § 3 is finding redundant paths in the given network that satisfy (8), i.e., are edge-disjoint. For that, we present an approach to derive all eligible paths $P_i \in \mathcal{P}$ connecting nodes M and S in the obtained topology graph $\mathcal{G}$ by leveraging both Menger's theorem [6] and the Max-Flow-Min-Cut theorem [10]. Menger states there are k pairwise edge-disjoint M-S paths if and only if S is still reachable from M after removing $k-1$ arbitrary edges from the graph [6]. Hence, we are interested in the minimum edge cut k that is required to disconnect the two nodes M and S in $\mathcal{G}$. By introducing the non-negative capacity function $c(e) = 1$, which assigns each edge $e_i \in E$ the maximum capacity of one, this problem is equivalent to maximizing the flow from M to S, as stated by the Max-Flow-Min-Cut theorem. Finally, the maximum flow can be efficiently computed by the Ford-Fulkerson algorithm [11] in polynomial time. Thus, to derive the number of edge-disjoint M-S paths in a given PTP network $\mathcal{G}$, we propose an adapted version of the Ford-Fulkerson algorithm which is depicted in Algorithm 1. First, we initialize the set of existing paths and the assigned flow values for each edge to zero (lines 1-2). Then, we repeatedly search for M-S paths containing only edges with remaining capacity, i.e., with $flow(e) = 0$ (line 3) and insert them into the set of existing paths (line 4). The search can be accomplished, for example, with the breadth-first search algorithm. For each found path, we additionally update the flow values of all comprising edges to mark their capacity as consumed (line 5) for subsequent search iterations. Finally, we return the resulting set of desired edge-disjoint paths $\mathcal{P}$ (line 6). By using an implementation based on the breadth-first-search, i.e., the Edmonds-Karp algorithm, the algorithm runs with an overall time complexity of $O(|V| \cdot |E|^2)$ [21].

6.4 Link Recommendation

As we know from § 4, each redundant path allows for one additional attacked link while maintaining the capabilities to detect and mitigate synchronization errors with our asymmetry calculations. From this insight, we can derive a design parameter to state certain security requirements for network planning. If we set the maximum number of attacked links n_{att} that we desire to tolerate in a network, e.g., derived from prior threat modeling, we can easily determine the required number of redundant paths n_{redun}. Particularly, we can state:

$$n_{redun} = 2 \cdot n_{att} + 1, \tag{18}$$

where n_{redun} also includes the synchronization path P_0. Furthermore, we specify the condition for sufficient security:

$$|\mathcal{P}| \geq n_{redun} \tag{19}$$

Algorithm 2 get_recommended_links

Input: Network $\mathcal{G} = (V, E)$, Source $M \in V$, Sink $S \in V$, Number of redundant M-S paths n_{redun}, Cost function $c(v_1, v_2) : V \times V \rightarrow \mathbb{R}$

Output: Updated set of edge-disjoint paths $\mathcal{P}$ from M to S, Set of edges $\mathcal{E}$ to be added

1: $\mathcal{P} \leftarrow find\_edge\_disjoint\_paths(\mathcal{G}, M, S)$ /* Initialize */
2: $\mathcal{E} \leftarrow \emptyset$
3: **if** $|\mathcal{P}| \geq n_{redun}$ **then** /* Terminate if condition is already met */
4: **return** $\mathcal{P}, \mathcal{E}$
5: **for each** $P_i \in \mathcal{P}$ **do** /* Delete all existing M-S paths */
6: **for each** $e_j \in P_i$ **do**
7: $E \leftarrow E \setminus \{e_j\}$
8: **while** $|\mathcal{P}| < n_{redun}$ **do**
9: $m\_reach \leftarrow reachable(\mathcal{G}, M)$ /* Get all nodes reachable from M and S */
10: $s\_reach \leftarrow reachable(\mathcal{G}, S)$
11: **if** $|m\_reach| = 0 \vee |s\_reach| = 0$ **then**
12: **break**
13: $new\_edge \leftarrow get\_best\_link(m\_reach, s\_reach, c)$ /* Get link with minimal cost */
14: $E \leftarrow E \cup \{new\_edge\}$
15: $P \leftarrow get\_path(M, S)$
16: $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$ /* Add result to output sets */
17: $\mathcal{E} \leftarrow \mathcal{E} \cup \{new\_edge\}$
18: **for each** $e_j \in P$ **do** /* Delete new M-S path */
19: $E \leftarrow E \setminus \{e_j\}$
20: **return** $\mathcal{P}, \mathcal{E}$

Once n_{redun} is known, we can analyze the network and check whether (19) is satisfied. If not, we can update the network accordingly. For that, we propose Algorithm 2, which outputs a set of recommended new links to increase the path redundancy and meet the criterion. First, we find all existent M-S paths $P_i \in \mathcal{P}$ with Algorithm 1 and initialize the set of newly recommended edges (lines 1-2). If $|\mathcal{P}|$ is sufficiently large, we can terminate here (lines 3-4). Otherwise, we remove the edges of all existing M-S paths (lines 5-7), get the remaining nodes that are still reachable from M and S respectively (lines 9-10), and determine the best link to add (line 13). Note that this selection depends on the chosen cost function $c(v_1, v_2)$, which quantifies the cost for a new edge between the nodes $v_1 \in V$ and $v_2 \in V$. For example, the physical distance between nodes can be a suitable cost function here. The recommended edge and the resulting M-S path are added to the output sets before the related edges are removed again from the graph for the next iteration (lines 18-19). This process is repeated until either enough new paths have been added or no further nodes are reachable from M and S anymore. Note that with Algorithm 2, it is possible to optimize certain network parts specifically. Hence, the most vulnerable sections can be protected more carefully if the network infrastructure is secured to varying degrees.

6.5 Traffic Routing

For both default PTP synchronization and the proposed asymmetry analysis as introduced in § 3, deterministic routing is important. Due to PTP's path symmetry assumption, the outward and return routes of PTP event messages must be identical even though hardware timestamping is used. The asymmetry analysis demands the additional condition that the measurement packets need to

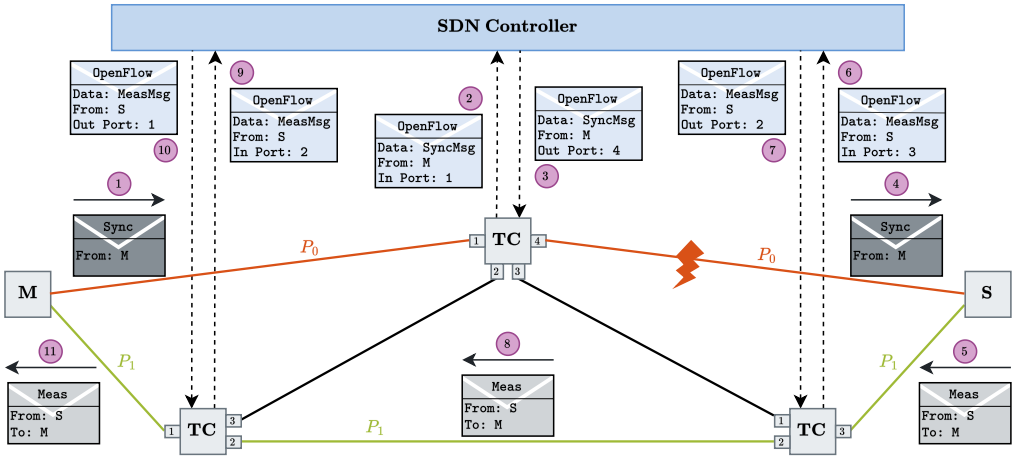


Fig. 7. Illustrating packet routing and SDN controller interaction for one asymmetry measurement cycle. When a new Sync message arrives at a switch (1), it is passed on to the SDN controller (2) to get the information on which output port to forward the packet (3). Once the Sync message arrives at the destination (4), it triggers Meas packets on all other ports in response (5). They follow the same procedure (6-10) until they finally return to the originator to complete the measurement cycle (11).

pass on a redundant path, i.e., a very specific route, in order to work. With the network topology obtained in § 6.2 and Algorithm 1, the SDN controller can calculate all redundant paths between the selected master and all slave nodes, determine optimal routes for all relevant packets, and enforce them once the switches ask for instructions. The routing process of an entire measurement cycle is shown in Fig. 7. As long as the topology does not change, the calculated routes are fixed, and appropriate flows can be set accordingly to improve the network performance. Since we intentionally maintain loops in the network for redundancy reasons, another important routing aspect is to prevent packets from circulating in loops. For this, the controller computes a minimum spanning tree of the network and uses this tree for the routing if loops are not specifically required (e.g., for default PTP messages or legacy traffic). The controller also filters the traffic for PTP messages so that other traffic is unaffected[4].

6.6 Continuous Monitoring

To remain responsive to dynamic topology changes (e.g., temporary link failures), the controller continuously monitors the network and reacts to changes accordingly. This includes periodically repeating the network discovery process and rerunning Algorithm 1 if necessary. As this process incurs some network management-related overhead, the interval should be maximized, considering the trade-off between topology freshness and overhead. Once the security criterion in (19) is not met anymore, the SDN controller can run known countermeasures or alert the network operator.

6.7 Multipoint Synchronization

In typical PTP deployments, many nodes synchronize to a single *Grandmaster* clock. Thus, we need to find redundant paths for all eligible synchronization paths. For this purpose, we apply all

[4]The SDN controller can follow specific strategies for other traffic or use a best-effort approach. The exact implementation is application-specific and beyond the scope of this article.

proposed concepts for every node, which shall be synchronized to the *Grandmaster*. Particularly, we run Algorithm 1 and Algorithm 2 for each slave to get the relevant redundant paths and adapt our routing accordingly.

7 Evaluation

We extensively evaluate the performance of our proposed PTPsec protocol in a hybrid fashion. To analyze general detection capabilities and realistic timing behavior, we perform various measurements under different attack scenarios on our hardware testbed. Additionally, we validate the SDN approach and PTPsec's general scalability in a software simulation.

7.1 Testing Environment

7.1.1 Hardware Setup. The hardware setup consists of two PCs serving as master and slave to be synchronized. Both machines are operating on *Ubuntu 20.04* and run an implementation of our PTPsec protocol, which is based on *linuxptp-v3.1.1*[5]. The two OCs use Intel i210 Network Interface Cards (NICs) for IEEE 1588-compliant hardware timestamping support and are connected with wired Ethernet connections. The synchronization path is interrupted by an attack device running a Data Plane Development Kit (DPDK) application to act as a transparent L2 switch. Moreover, it exploits its Machine-in-the-Middle (MitM) position to deliberately delay either `Sync` or `Delay_Req` messages to implement an effective time delay attack against PTP, as described in § 2.2. The attacker also uses Intel i210 NICs to perform hardware timestamping. Like a TC, it captures both ingress and egress timestamps for event messages and updates the correction field accordingly to increase synchronization performance[6]. In contrast to a TC, the malicious delay is not compensated with this mechanism. Otherwise, the attack would not have any effect. The other path is a direct point-to-point connection and constitutes a redundant path considered genuine and symmetric for all following experiments. Extending the setup with more redundant paths follows the same principle, as illustrated in Fig. 8. To monitor the actual clock offset between master and slave, we compare the generated Pulse-Per-Second (PPS) signal phases on a Rohde&Schwarz RTB oscilloscope.

7.1.2 Software Simulation. For the software simulation, we run a virtual network with *Mininet v2.3*[7] that simulates the network depicted in Fig. 9. The virtual hosts run our PTPsec protocol implementation, which is similar to the one used in the hardware setup. However, we have to run in software timestamping mode due to the simulation, which leads to significant performance degradation. We use *os-ken v2.8.1*[8] for the SDN controller implementation. The attack is realized with *Mininet*'s `TCLink` class that allows for individual link delays. All experiments were conducted on an *Arch Linux* implementation using *Linux Kernel 6.12.8-arch1-1* running on an *Intel i5 10210U* processor and *16 GB* of RAM.

7.2 Asymmetry Detection

In total, we present four hardware experiments to validate the detection performance of our protocol. We attack the two relevant PTP event messages `Sync` and `Delay_Req` and assume that `Meas` and `Meas_Fup` are only sent via the symmetric redundant path.

7.2.1 Static Delay (`Sync`*).* In the first experiment, we start with a static attack scenario, where the attacker adds a constant delay of $\epsilon_1 = 50\,\mu s$ to all passing `Sync` messages. The attack starts

[5]https://github.com/richardcochran/linuxptp

[6]Technically, this step requires internal protocol access, which contradicts our chosen attacker model. Nevertheless, we only use it for accuracy improvements to emphasize the effectiveness of our method and not to perform the actual attack.

[7]https://github.com/mininet/mininet

[8]https://github.com/openstack/os-ken

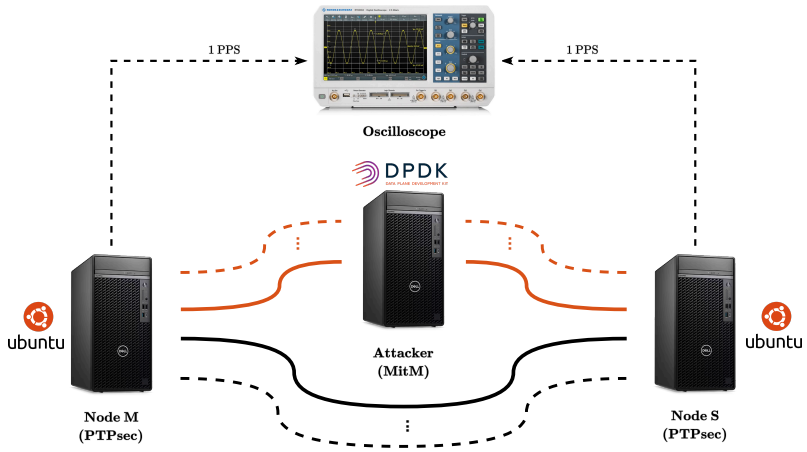


Fig. 8. Hardware testbed for comprehensively evaluating our proposed approach under realistic circumstances. The synchronization path is interrupted by a MitM attacker delaying selected PTP event messages in transit to diverge the slave clock. We closely monitor the malicious clock offset with the oscilloscope by comparing the generated PPS signals.

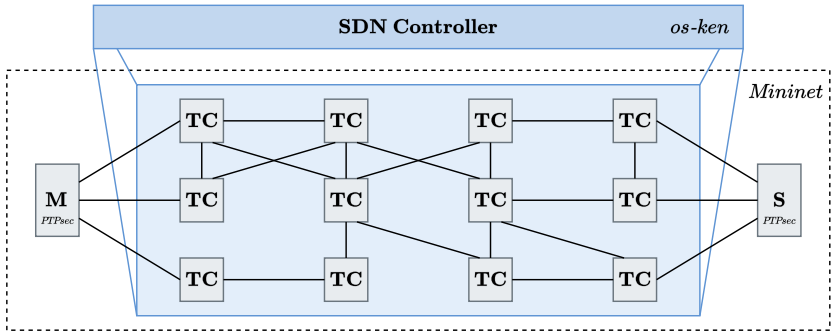
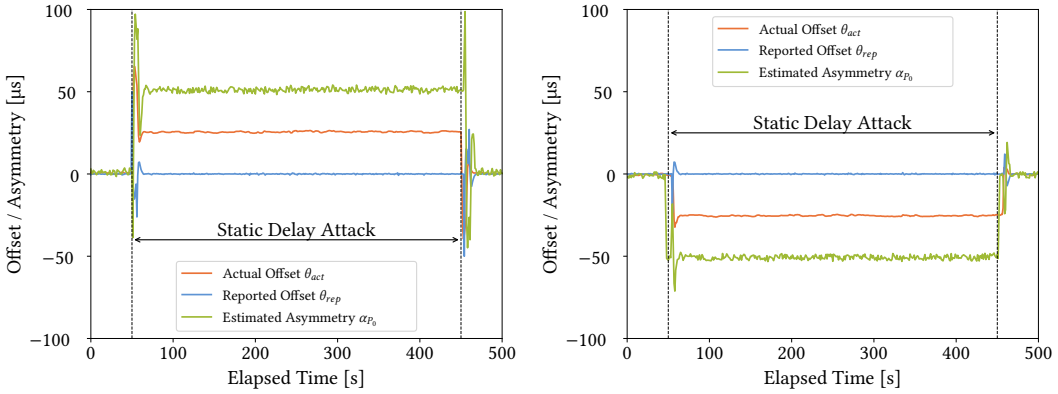


Fig. 9. Software simulation environment used to evaluate PTPsec's scalability. The depicted network is implemented in *Mininet* and orchestrated by an *os-ken* SDN controller implementation.

at $t = 50$ s and lasts until $t = 450$ s, as shown in Fig. 10a. During that period, we observe that the actual clock offset increases to $\theta_{act} = 25\,\mu s$, i.e., by half the introduced one-way delay, while the reported PTP clock offset θ_{rep} remains at zero. This behavior clearly shows the success of the delay attack because the actual clock offset changes to the expected value given by (2) without being reported by conventional PTP. Furthermore, the estimated synchronization path asymmetry rises to $\alpha_{P_0} = 50\,\mu s$ during the attack period. Since $\alpha_{P_0} > 0$, the path delay from node M to node S is higher than in the opposite direction, and we conclude an ongoing delay attack targeting the Sync message. Hence, the applied attack was successfully detected.

7.2.2 Static Delay (`Delay_Req`*).* We repeat the previous attack in the second experiment with the only difference in the targeted PTP message. This time, the attacker delays passing `Delay_Req` messages instead, which results in a path asymmetry in the opposite direction. As Fig. 10b depicts, the reported PTP clock offset θ_{rep} remains at zero while the actual clock offset $\theta_{act} = -25\,\mu s$ and



(a) Delay PTP Sync message ($\epsilon_1 = 50\,\mu s$). (b) Delay PTP Delay_Req message ($\epsilon_2 = 50\,\mu s$).

Fig. 10. Static delay attack between $t_1 = 50\,s$ and $t_2 = 450\,s$ targeting the two PTP event messages Sync (a) and Delay_Req (b). The difference between the measured clock offset (orange) and the reported clock offset (blue) confirms the ongoing attacks. Nevertheless, our proposed asymmetry analysis successfully detects this malicious behavior (green) in both scenarios.

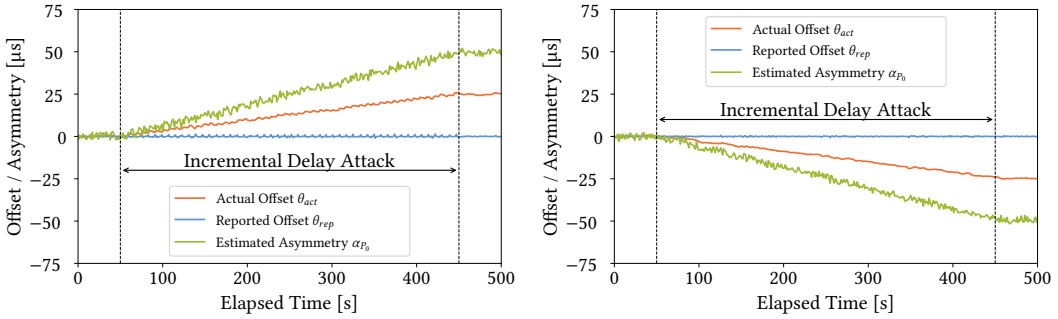
the estimated path asymmetry $\alpha_{P_0} = -50\,\mu s$ both show a change of sign. The negative values match our expectations due to the different PTP message that was targeted. These results confirm that the attack detection works independently of the present asymmetry direction.

*7.2.3 Incremental Delay Attack (*Sync*).* Next, we change the attack method to an incremental delay attack to evaluate the detection performance dynamically. Therefore, we gradually increase the packet delay for passing Sync messages to slowly diverge the slave's clock. The attack starts at $t = 50\,s$. Each second, we increase the added delay by $\Delta\epsilon = 125\,ns$ to eventually reach a level of $\epsilon_1 = 50\,\mu s$ at $t = 450\,s$ which remains as a static offset until the end of the experiment. From the results in Fig. 11a, we observe that not only does the actual clock offset gradually ascend from $\theta_{act} = 0\,\mu s$ to $\theta_{act} = 25\,\mu s$, but also the estimated path asymmetry α_{P_0} slowly follows the incremental delay to reach a final level of $\alpha_{P_0} = 50\,\mu s$.

*7.2.4 Incremental Delay Attack (*Delay_Req*).* Finally, we repeat the incremental delay attack targeting Delay_Req messages. During the attack period, the actual clock offset and the estimated path asymmetry descend to $\theta_{act} = -25\,\mu s$ and $\alpha_{P_0} = -50\,\mu s$, respectively, attesting similar results albeit with the opposite sign. Fig. 11b displays the results. Therefore, our proposed protocol also reliably detects incremental delay attacks independent of the targeted PTP message.

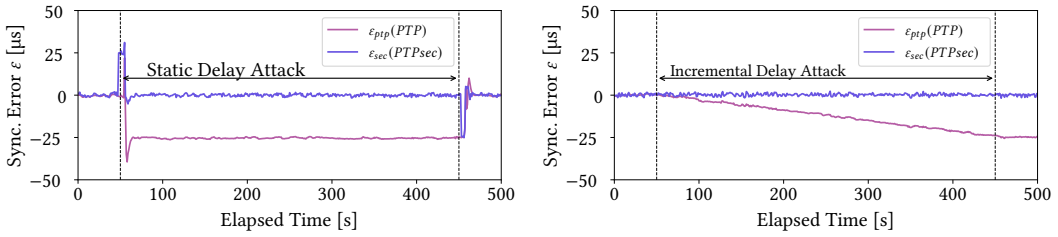
7.3 Attack Mitigation

As the previous experiments show, our proposed PTPsec protocol reliably detects the malicious path asymmetry in all attack scenarios. To further evaluate the attack mitigation performance, we analyze the synchronization errors ε_{ptp} and ε_{sec} of both conventional PTP and PTPsec, respectively, by comparing the reported clock offset to the actual clock offset measured by the oscilloscope. For PTPsec, we use the rectified offset from (14). As the results in Fig. 12a and Fig. 12b illustrate, PTPsec reliably mitigates the attacks. The overshooting at the start and the end of the static attack results from the sudden change in delay that cannot be immediately followed by PTP's control algorithm. In comparison, we cannot observe such a behavior for incremental attacks because of the smoother change in delay over time.



(a) Delay PTP Sync message ($0\,\mu s \leq \epsilon_1 \leq 50\,\mu s$) (b) Delay PTP `Delay_Req` message ($0\,\mu s \leq \epsilon_2 \leq 50\,\mu s$)

Fig. 11. Incremental delay attack between $t_1 = 50$ s and $t_2 = 450$ s targeting both PTP event messages Sync (a) and `Delay_Req` (b). The delay is gradually increased by $\Delta\epsilon = 125$ ns per second starting from $\epsilon_1 = \epsilon_2 = 0\,\mu s$, and then transitioned to a static attack at the final values of $\epsilon_1 = 50\,\mu s = \epsilon_2 = 50\,\mu s$. The difference between the measured clock offset (orange) and the reported clock offset (blue) confirms the ongoing attack. Nevertheless, our proposed asymmetry analysis successfully detects this malicious behavior (green).



(a) Static Delay Attack ($\epsilon_2 = 50\,\mu s$). (b) Incremental Delay Attack ($0\,\mu s \leq \epsilon_2 \leq 50\,\mu s$).

Fig. 12. Remaining clock error after delay attack compensation considering the estimated path asymmetry for static (a) and incremental (b) delay attacks.

7.4 Detection Time

In addition to the general detection and mitigation capabilities, we further examine the timing behavior of the proposed approach. Mainly, we investigate the elapsed time between the start of the attack and its visibility in the measured path asymmetry, i.e., the detection time of our method. This time is critical since it potentially opens a small window for attackers during which the deployed attack is effective without an appropriate system response and, thus, needs to be minimized. In our setup, the Sync message timeout interval is set to 1 s, which also predetermines the clock update and asymmetry measurement rate to approximately this period. As the plot in Fig. 13 shows, the measured path asymmetry follows the actual clock offset almost immediately within less than five time steps at the start and the end of the attack.

7.5 Scalability

Finally, we evaluate the SDN application and its performance in facilitating PTPsec in more extensive networks. For that, we perform two measurements within the simulation environment as introduced in § 7.1.2. The first experiment comprises a static delay attack targeting the PTP Sync message with a delay of $\epsilon_1 = 20$ ms. As Fig. 14a illustrates, our asymmetry analysis yields an average estimation $\alpha_{P_0} \approx 19$ ms (green), which is close to the actual path asymmetry (red). In the second experiment, we

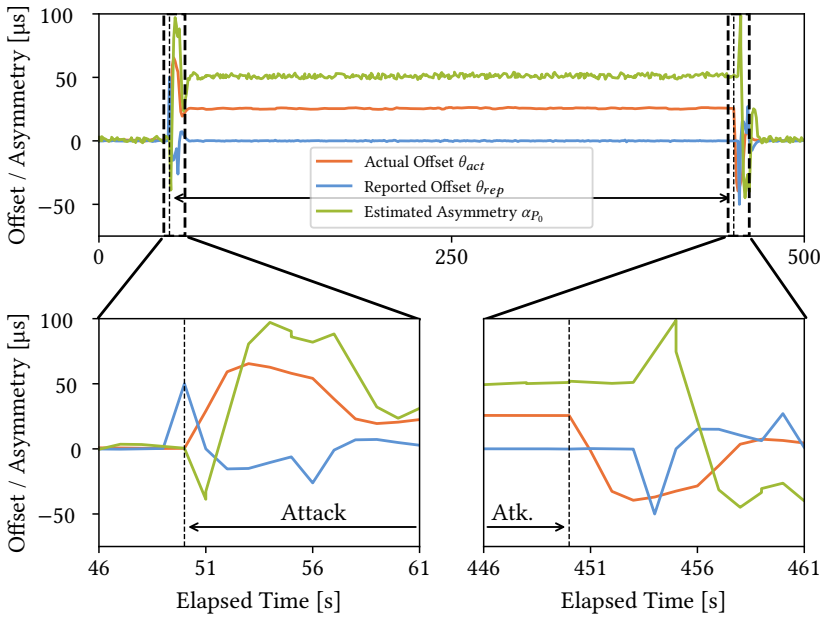
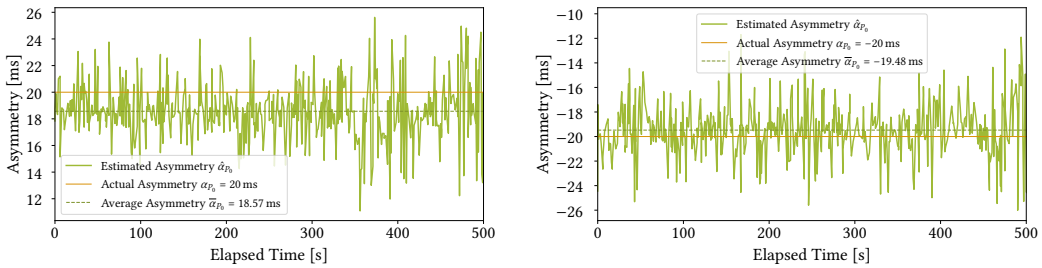


Fig. 13. Timing analysis of our attack detection approach. The magnified plots at the bottom illustrate our method's fast detection time at the applied attack's start and end.



(a) Delay PTP Sync message ($\epsilon_1 = 20$ ms)

(b) Delay PTP `Delay_Req` message ($\epsilon_2 = 20$ ms)

Fig. 14. Static delay attack targeting the two PTP event messages Sync (a) and `Delay_Req` (b) in our SDN-based simulation environment. Our analysis successfully reveals the introduced asymmetry in both cases. The jitter originates from the software timestamping method that cannot compensate for the delays induced by the switches and the SDN controller communication.

repeat the attack targeting the PTP `Delay_Req` message, again with a delay of $\epsilon_2 = 20$ ms. Similarly, our analysis results in an average estimation $\alpha_{P_0} \approx -19$ ms (green) slightly above the actual path asymmetry (red) as depicted in Fig. 14b. Note that PTPsec uses software timestamping here due to the simulation environment which cannot compensate for the variable delays introduced by the switches and the SDN controller. Consequently, the measurements include much more jitter compared to our hardware experiments while still detecting the attacks successfully.

8 Discussion

The proposed PTPsec protocol has successfully detected and mitigated the delay attacks in all conducted experiments, validating the effectiveness of our approach. Since the chosen scenarios represent all known time delay attack strategies, we can conclude that our cyclic asymmetry analysis enables reliable attack detection and mitigation, provided redundant paths are available in the network. The presented SDN approach successfully finds the required redundant paths in arbitrary networks. It even recommends adding new links to meet the security requirements regarding the expected number of attacked links. Furthermore, it provides the required routing capabilities to deterministically route the default PTP messages as well as the newly introduced `Meas` and `Meas_Fup` messages along with other network traffic. While maintaining redundancy in networks is costly, it increases reliability and, in this case, also security, which is usually not free. Interestingly, this redundancy requirement is already enforced by some communication standards, such as TSN. So, our secure protocol can be deployed in related applications at no cost. Regarding the protocol overhead of PTPsec, the increased security also incurs some costs. Since both the number of successfully detected asymmetric paths and the required packet count scale with the number of redundant network paths, it is a trade-off between desired security and acceptable network load that system designers have to make. The additional SDN traffic does not affect the synchronization performance if hardware timestamping is used. However, the SDN controller introduces some computational overhead depending on the network size and the frequency of changes. While the routing runs continuously, the algorithms for topology discovery and finding redundant paths are executed at each alteration with the related time complexities that grow with increasing network size.

9 Related Work

Research has broadly covered attack detection and prevention in time synchronization in recent years. The proposed solutions to protect PTP from time delay attacks can be mainly clustered into three groups. First, there exist threshold-based detection techniques. The idea is to continuously monitor the reported clock offset and path delay and decide whether or not the system is under attack based on a defined threshold. The threshold can be either static, as proposed by Ullmann et al. [35], or dynamically updated over time, as presented in [22, 36]. Li et al. introduce a threshold-based attack detection approach, focusing on the frequency differences instead [23]. In [12], the author monitors the clock servo's Kalman filter to derive potential attack detection criteria. However, the performance of these techniques strongly depends on the chosen threshold, which is quite challenging. The second category comprises solutions that rely on special guard devices and additional reporting systems deployed in the network. Alghamdi et al. [2] present the idea of a trusted supervisor node that performs anomaly detection in the network to detect ongoing attacks. Moussa et al. [30] propose a similar idea, including a dedicated guard node that participates in the time synchronization protocol but does not update its clock. Instead, it only compares the results to another time reference based on Global Navigation Satellite Systems (GNSSs) to reveal potential delay attacks. However, the guard node only secures its synchronization path, and other nodes not sharing the same path are still vulnerable. In [31], Moussa et al. refine their initial proposal and introduce a more sophisticated approach, where all nodes additionally send timestamped reports to a specific network time reference node. These reports are exchanged through custom event messages unrelated to the PTP protocol stack. Thus, sophisticated attackers can distinguish report messages from actual PTP messages and react differently to disrupt time synchronization without noticing. Moradi et al. present similar methods [28, 29], including dedicated reporting schemes for attack detection. The third class of countermeasures utilizes path redundancy for time delay

Table 1. Delay Attack Detection Approaches Comparison

Solution	Year	Approach	Attack Resistance			Scal.	Appl.	Exp.		Acc.	OSS
			Static	Incr.	Select.			SW	HW		
Yang et al. [36]	2013	Threshold	●	◑	●	○	○	●	○	≈10 µs	○
Ullmann et al. [35]	2009	Threshold	●	◑	●	○	○	○	○	-	○
Li et al. [22]	2021	Threshold	●	◑	●	◑	●	○	●	≈10 µs	○
Li et al. [23]	2023	Threshold	●	○	●	●	○	●	●	≈1 ns	○
Giorgi et al. [12]	2024	Threshold	●	◑	●	●	◑	●	○	≈100 µs	○
Moradi et al. [29]	2024	Guard	●	●	○	●	◑	●	○	≈1 ms	○
Alghamdi et al. [2]	2020	Guard	●	●	○	○	○	○	○	-	○
Moussa et al. [30]	2015	Guard	●	●	●	○	◑	●	○	-	○
Moussa et al. [31]	2020	Guard	●	●	○	●	◑	●	○	≈1 ms	○
Moradi et al. [28]	2021	Guard	●	●	○	●	◑	●	○	≈1 ms	○
Mizrahi et al. [26]	2012	Path Red.	●	●	●	●	○	○	○	-	○
Neyer et al. [32]	2019	Path Red.	●	●	●	○	○	○	●	≈100 µs	○
PTPsec	**2024**	**Path Red.**	●	●	●	●	●	●	●	**≈10 µs**	●

○ no or n/a, ◑ partially, ● yes

attack protection. Some works, such as [20, 27, 34], cover path redundancy for network-based time synchronization to improve the provided synchronization accuracy. While not explicitly mentioning any security aspects, the works already indicate the capabilities of redundant path approaches, which could also be adopted in the security domain. In [26], Mizrahi proposes redundant paths as a countermeasure for delay attacks and presents a theoretical analysis of this idea. Furthermore, Neyer et al. [32] perform supplementary experiments on a hardware testbed to showcase the general applicability of redundant paths as viable improvement for time synchronization security.

Comparative Analysis: Table 1 presents an additional comparison between our proposed solution and existing works. The comparison is based on the year of publication, the adopted approach, the ability to detect static, incremental, and selective delay attacks, the scalability of the proposed solution, its applicability for actual PTP deployments, the experimental validation method employed (hardware setup, software simulation, or no validation), the detection accuracy achieved in the experiments, and whether an Open-Source Software (OSS) implementation of the proposed solution exists. While all the solutions can detect static attacks, [12, 22, 23, 35, 36] fall short of fully detecting incremental attacks due to the threshold-based approach. Others, like [2, 28, 29, 31], are still vulnerable to selective delay attacks, where the attackers distinguish between PTP and other network traffic. In contrast to PTPsec, many other solutions do not adequately scale either due to the need for a recurring setup phase after each network change, as seen in [35, 36], including single points of failure [2, 30], or due to limitations in the system model [32]. Furthermore, hardware testbed validation is essential for ensuring the effectiveness of proposed detection methods, which most of the compared solutions lack. None of the other works provide open-source implementations of their solutions. Based on Table 1, the three most comparable solutions to our system are [22], [26], and [32]. However, [22] cannot reliably detect incremental delay attacks because of the threshold-based approach while additionally requiring an initial learning phase that becomes invalid with any network change. Furthermore, the presented solutions in [26] and [32] lack experimental validation and scalability, respectively. Moreover, while partially providing sound detection approaches, none of the compared works can mitigate time delay attacks, which is an exclusive feature of PTPsec.

10 Conclusion

This work introduces a theoretical model for cyclic path asymmetry analysis that can be efficiently used for time delay attack detection and mitigation. The derived detection method reliably reveals both static and incremental time delay attacks provided at least one redundant network path with symmetric delay exists. With the introduced SDN approach, we can maintain a live state of the network topology, find redundant paths, recommend new links to increase path redundancy, and ensure deterministic routing. With PTPsec, we present a protocol that secures the latest IEEE 1588 standard against time delay attacks by incorporating our proposed mitigation techniques on top of conventional cryptographic countermeasures. This makes PTPsec fully resilient against all known attacks against PTP. The experimental results show that our approach successfully identifies fine-grained asymmetry changes with microsecond accuracy and minimal detection time in all evaluated scenarios. Furthermore, we verify the scalability of our approach in simulation. The authors have provided public access to their code and data at https://github.com/tum-esi/ptpsec.

Acknowledgments

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of “Souverän. Digital. Vernetzt.”. Joint project 6G-life, project identification number: 16KISK002.
This work is supported by the European Union-funded project CyberSecDome (Agreement No.: 101120779).

References

[1] Waleed Alghamdi and Michael Schukat. 2017. Advanced methodologies to deter internal attacks in PTP time synchronization networks. In *2017 28th Irish Signals and Systems Conference (ISSC)*. IEEE, Killarney, Ireland, 1–6.
[2] Waleed Alghamdi and Michael Schukat. 2020. Cyber Attacks on Precision Time Protocol Networks—A Case Study. *Electronics* 9, 9 (2020), 1398.
[3] Waleed Alghamdi and Michael Schukat. 2021. Precision time protocol attack strategies and their resistance to existing security extensions. *Cybersecurity* 4, 1 (2021), 1–17.
[4] Robert Annessi, Joachim Fabini, Felix Iglesias, and Tanja Zseby. 2018. Encryption is Futile: Delay Attacks on High-Precision Clock Synchronization. *arXiv preprint arXiv:1811.08569* (2018).
[5] Sergio Barreto, Aswin Suresh, and Jean-Yves Le Boudec. 2016. Cyber-attack on packet-based time synchronization protocols: The undetectable Delay Box. In *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*. IEEE, Taipei, Taiwan, 1–6.
[6] Thomas Böhme, Frank Göring, and Jochen Harant. 2001. Menger’s theorem. *Journal of Graph Theory* 37, 1 (2001), 35–36.
[7] Jin-Hee Choi and Chuck Yoo. 2005. One-way delay estimation and its application. *Computer Communications* 28, 7 (2005), 819–828.
[8] Andreas Finkenzeller, Oliver Butowski, Emanuel Regnath, Mohammad Hamad, and Sebastian Steinhorst. 2024. PTPsec: Securing the Precision Time Protocol Against Time Delay Attacks Using Cyclic Path Asymmetry Analysis. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*. IEEE, Vancouver, BC, Canada, 461–470.
[9] Andreas Finkenzeller, Thomas Wakim, Mohammad Hamad, and Sebastian Steinhorst. 2022. Feasible Time Delay Attacks Against the Precision Time Protocol. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, Rio de Janeiro, Brazil, 3375–3380.
[10] Lester Randolph Ford and Delbert R Fulkerson. 1956. Maximal flow through a network. *Canadian journal of Mathematics* 8 (1956), 399–404.
[11] Lester Randolph Ford Jr and Delbert Ray Fulkerson. 2015. *Flows in networks*. Vol. 54. Princeton university press.
[12] Giada Giorgi. 2024. Measurable quantities in a synchronization system under attack: A first step to implement a detection system. In *2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, Glasgow, United Kingdom, 1–6.
[13] O. Gurewitz, I. Cidon, and M. Sidi. 2006. One-way delay estimation using network-wide measurements. *IEEE Transactions on Information Theory* 52, 6 (2006), 2710–2724.

[14] Omer Gurewitz and Moshe Sidi. 2001. Estimating one-way delays from cyclic-path delay measurements. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, Vol. 2. IEEE, Anchorage, AK, USA, 1038–1044.

[15] 2020. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)* (2020), 1–499.

[16] 2018. IEEE/IEC International Standard - Measuring relays and protection equipment - Part 118-1: Synchrophasor for power systems - Measurements. *IEC/IEEE 60255-118-1:2018* (2018), 1–78.

[17] 2016. IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery. *IEEE Std 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009)* (2016), 1–146.

[18] 2020. IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)* (2020), 1–421.

[19] Keith Kirkpatrick. 2013. Software-defined networking. *Commun. ACM* 56, 9 (2013), 16–19.

[20] Andre Komes and Cristian Marinescu. 2012. IEEE 1588 for redundant ethernet networks. In *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*. IEEE, San Francisco, CA, USA, 1–6.

[21] Peter Lammich and S Reza Sefidgar. 2016. Formalizing the edmonds-karp algorithm. In *Interactive Theorem Proving: 7th International Conference, ITP 2016, Nancy, France, August 22-25, 2016, Proceedings 7*. Springer, Cham, 219–234.

[22] Hongxing Li, Dengkui Li, Xiaodong Zhang, Guochu Shou, Yihong Hu, and Yaqiong Liu. 2021. A security management architecture for time synchronization towards high precision networks. *IEEE Access* 9 (2021), 117542–117553.

[23] Yang Li, Jinlong Hu, Yan Pan, Wei Huang, Li Ma, Jie Yang, Shuai Zhang, Yujie Luo, Chuang Zhou, Chenlin Zhang, et al. 2023. Secure two-way fiber-optic time transfer against sub-ns asymmetric delay attack with clock model-based detection and mitigation scheme. *IEEE Transactions on Instrumentation and Measurement* 72 (2023), 1–14.

[24] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review* 38, 2 (2008), 69–74.

[25] Chetan Mishra, Luigi Vanfretti, Jaime Delaree Jr, and Kevin D Jones. 2024. Internal clock errors in synchrophasor ambient data: Effects, detection, and a posteriori estimation-based correction. *International Journal of Electrical Power & Energy Systems* 161 (2024), 110208.

[26] Tal Mizrahi. 2012. 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, San Francisco, CA, USA, 1–6.

[27] Tal Mizrahi. 2012. Slave diversity: Using multiple paths to improve the accuracy of clock synchronization protocols. In *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*. IEEE, San Francisco, CA, USA, 1–6.

[28] Mohsen Moradi and Amir Hossein Jahangir. 2021. A new delay attack detection algorithm for PTP network in power substation. *International Journal of Electrical Power & Energy Systems* 133 (2021), 107226.

[29] Mohsen Moradi and Amir Hossein Jahangir. 2024. A Petri net model for Time-Delay Attack detection in Precision Time Protocol-based networks. *IET Cyber-Physical Systems: Theory & Applications* 9, 4 (2024), 407–423.

[30] Bassam Moussa, Mourad Debbabi, and Chadi Assi. 2015. 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, Miami, FL, USA, 497–502.

[31] Bassam Moussa, Marthe Kassouf, Rachid Hadjidj, Mourad Debbabi, and Chadi Assi. 2020. An Extension to the Precision Time Protocol (PTP) to Enable the Detection of Cyber Attacks. *IEEE Transactions on Industrial Informatics* 16, 1 (2020), 18–27.

[32] Johannes Neyer, Lukas Gassner, and Cristian Marinescu. 2019. Redundant Schemes or How to Counter the Delay Attack on Time Synchronization Protocols. In *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. IEEE, Portland, OR, USA, 1–6.

[33] Arman Sargolzaei, Kang Yen, and Mohamed N Abdelghani. 2014. Delayed inputs attack on load frequency control in smart grid. In *ISGT 2014*. IEEE, Washington, DC, USA, 1–5.

[34] Alexander Shpiner, Yoram Revah, and Tal Mizrahi. 2013. Multi-path time protocols. In *2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS) Proceedings*. IEEE, Lemgo, Germany, 1–6.

[35] Markus Ullmann and Matthias Vogeler. 2009. Delay attacks — Implication on NTP and PTP time synchronization. In *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, Brescia, Italy, 1–6.

[36] Qingyu Yang, Dou An, and Wei Yu. 2013. On time desynchronization attack against IEEE 1588 protocol in power grid systems. In *2013 IEEE Energytech*. IEEE, Cleveland, OH, USA, 1–5.