Feasible Time Delay Attacks Against the Precision Time Protocol

Andreas Finkenzeller, Thomas Wakim, Mohammad Hamad, Sebastian Steinhorst
Department of Electrical and Computer Engineering
Technical University of Munich, Germany
Email: firstname.lastname@tum.de

***Abstract*—Time synchronization in packet-switched networks has evolved into an indispensable prerequisite for many modern applications. In addition to high accuracy demands, also reliability and security are evermore of great concern. Despite the proposal of supplementary security concepts in the past years such as the four prongs in Annex P of the IEEE 1588 standard, available protocols are still vulnerable to time delay attacks. In this paper, we propose multiple methods to implement realistic delay attacks and verify the feasibility on a hardware testbed. Furthermore, we perform a risk analysis to evaluate the actual threat of delay attacks in practical applications. Our analysis shows that time delay attacks still pose a great threat to current systems and further research is required to find sound countermeasures.**

***Index Terms*—IEEE 1588, PTP, Delay Attack, Time Synchronization, Security**

I. Introduction

Reliable time synchronization has become a very important requirement in modern applications over the last years. Smart Grids, 5G backhaul networks, and Industry 4.0 are only some examples that rely on precise clock synchronization to improve performance. Besides the established synchronization method via Global Navigation Satellite Systems (GNSS), time synchronization protocols over packet-switched networks are getting more commonly used to accomplish this task. A popular protocol is, for example, the Precision Time Protocol (PTP) that is also standardized as IEEE 1588 [1]. One main advantage of PTP compared to GNSS is the increased flexibility due to the independence from satellite signals. Hence, also indoor deployments are fully supported. Furthermore, many systems already include network connectivity and thus do not require additional hardware. However, PTP relies on suitable network configurations as the protocol makes further assumptions, e.g., on the network path delay, to work properly. This introduces new threats into the system, especially when considering the highly increasing number of malicious cyber attacks over the last years. Initially, time synchronization protocols were developed without security concerns in mind and, thus, have been proven to be vulnerable to many different attacks, such as message manipulation [2], message dropping and insertion [3], Denial-of-Service (DoS) [4], master node falsification [5], and delay attacks [6]. As a response, PTP was updated accordingly and now includes additional countermeasures which are stated in the four-pronged approach of Annex P. This approach mainly ensures message integrity and authenticity to mitigate the aforementioned attacks. Message confidentiality could also be ensured but is usually not necessary since time information is publicly available. Although these changes can prevent most of the attacks, especially delay attacks remain an unresolved problem. There exist many studies about the theoretical principles of delay attacks [6]–[8] against PTP, whereas, to the best of our knowledge, practical delay attack methods with implementation on realistic testbeds including an appropriate risk analysis have not been extensively discussed in literature.

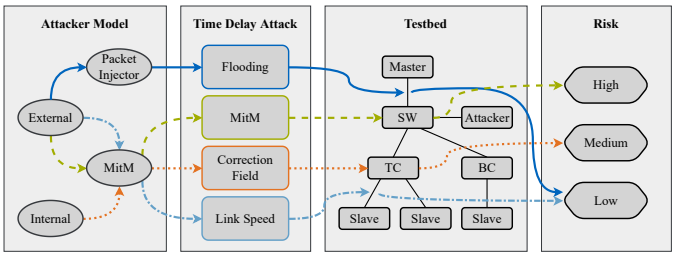


Fig. 1: Threat model of our proposed implementation methods for time delay attacks against PTP. The analysis shows that the prong-based security concept in the newest standard still imposes a high risk to all applications which adopt the time synchronization protocol.

In this paper, we analyze the feasibility of practical delay attack implementations against PTP by means of different strategies and additionally present an appropriate threat model (Sec. II) as visualized in Fig. 1. In particular, we

- show different strategies for practical delay attack implementations against PTP (Sec. III),
- present an open-source testbed[1] with which we verify the feasibility of the proposed strategies (Sec. IV),
- conduct a risk analysis that expounds the severity for current systems and exposes the urgent need for solutions (Sec. V).

II. System & Threat Model

A. Time Synchronization Principles

The goal of all time synchronization protocols is to minimize the offset θ between two or more clocks with the exchange of a set of messages as depicted in Fig. 2a by means of PTP. By selecting one clock as the time reference, all measured timestamps in the system can be expressed according to this reference clock as exemplified in the following equation system based on two clocks

$$t_1 = t_0 + \theta + d_1 \tag{1}$$
$$t_3 = t_2 - \theta + d_2 \tag{2}$$

978-1-6654-3540-6/22/$31.00 ©2022 European Union

[1]Refer to https://github.com/tum-esi/time-sync-testbed for the source code.

where t_0, t_1, t_2, and t_3 are the captured timestamps, θ the clock offset to the reference clock, and d_1 and d_2 the individual link delays. The four timestamps are the only known system parameters, which results in an underconstrained system that has, in general, no unique solution, i.e., the clocks cannot be synchronized. Even with the exchange of further messages, we cannot resolve this problem since this is an inherent problem in packet-switched networks as discussed in [9]. Nevertheless, with the additional assumption

$$d_1 = d_2 = d, \tag{3}$$

which means that both link delays are equal and thus the network path is symmetric, we have another equation and can determine the clock offset uniquely. Under normal conditions, a symmetric path delay is a valid assumption that leads to the following formulas for θ and the (symmetric) link delay d:

$$\theta = \frac{(t_1 - t_0) - (t_3 - t_2)}{2} \tag{4}$$

$$d = \frac{(t_1 - t_0) + (t_3 - t_2)}{2} \tag{5}$$

However, if this assumption breaks due to malicious attacks, for example, when the attacker is able to thwart the clock synchronization or even set an arbitrary time if the delay asymmetry is controllable. Although this principle holds for all time synchronization protocols, we refer only to PTP in the remainder of this work as all results can be transferred to other protocols accordingly.

B. Precision Time Protocol

The IEEE 1588 standard [1], also known as PTP, is a clock synchronization protocol that is based on the aforementioned principles. It includes a Master-Slave[2] communication architecture where many slaves synchronize their clocks to a selected master clock, the so-called Grandmaster. This Grandmaster is selected within a negotiation process among all participating nodes by means of the Best Master Clock Algorithm (BMCA) that happens prior to the actual synchronization. Afterwards, master and slaves periodically exchange several messages and save the according timestamps whenever the packet is sent or received. Fig. 2a depicts one entire communication cycle. With the use of special Network Card Interfaces (NICs) that are capable of capturing the timestamps directly in hardware, the synchronization accuracy can be significantly increased. The achieved decoupling of varying delays in the software stack from the actual link delay enables nanosecond accuracies between two hops. In order to preserve this high accuracy also over multi-hop networks, all intermediary nodes must also support PTP and provide hardware timestamping capabilities in particular[3]. The standard thereby differentiates two different device types, i.e., Transparent Clocks (TCs) and Boundary Clocks (BCs). A TC acts like a transparent switch because it measures the residence time of the packets and adds it to a reserved Correction Field (CF) so that this extra delay

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

[3]Using legacy devices without appropriate support is still possible but will degrade the accuracy.

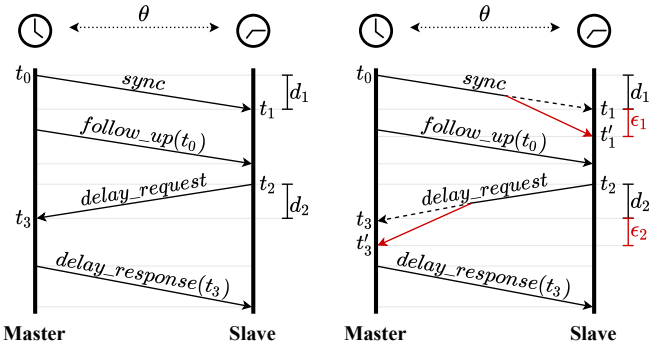


Fig. 2: PTP message flow with timestamping to minimize the clock offset θ (a). When the attackers are able to delay either the *snyc* or *delay_request* message (b), they can introduce asymmetric path delays that will break the synchronization.

can be considered in the slave's offset calculation. A BC, in contrast, constitutes an endpoint in the network and acts as new master for all following devices in a subnet while still synchronizing itself to the Grandmaster. The delay calculation can additionally run in two modes, namely End-to-End (E2E) or Peer-to-Peer (P2P) mode. The delays are either measured for each individual link (P2P) or over the full path between master and slave (E2E).

In the recent version of the standard from 2019 [1], a new security concept has been added in Annex P as an informative supplement that contains four prongs with general security advises. These mainly cover cryptographic security mechanisms included in the protocol (Prong A) and in external transport mechanisms (Prong B) as well as architectural security mechanisms (Prong C) and monitoring (Prong D) which are not further specified.

C. Time Delay Attack

Time synchronization protocols heavily rely on the symmetric path delay assumption, which we have already discussed in Sec. II-A. Hence, if attackers are able to break this assumption in any way, they can thwart the synchronization. The clock offset in Eq. 4 is calculated with four timestamps that can be potentially attacked. However, merely t_1 and t_3 are captured after a packet transmission and thus relevant for the attack as only these timestamps can be affected by the attacker with a packet delay. The interesting packets are, therefore, the `sync` and the `delay_request` messages.

Let's assume, the attacker can delay both messages by arbitrary delays ϵ_1 and ϵ_2 ($\epsilon_1 \neq \epsilon_2$)[4] as depicted in Fig. 2b so that Eq. 1 and Eq. 2 change to:

$$t_1' = t_0 + \theta + d_1 + \epsilon_1 \tag{6}$$

$$t_3' = t_2 - \theta + d_2 + \epsilon_2 \tag{7}$$

With the assumption from Eq. 3, we conclude

[4]If $\epsilon_1 = \epsilon_2$, we would end up with symmetric delay again, which is the original assumption we want to break.

$$\theta' = \frac{(t_1 - t_0) - (t_3 - t_2)}{2} - \frac{\epsilon_1 - \epsilon_2}{2} \tag{8}$$

$$d' = \frac{(t_1 - t_0) + (t_3 - t_2)}{2} - \frac{\epsilon_1 + \epsilon_2}{2} \tag{9}$$

and the following relationship accordingly:

$$\theta' = \theta - \frac{\epsilon_1 - \epsilon_2}{2} \tag{10}$$

$$d' = d - \frac{\epsilon_1 + \epsilon_2}{2} \tag{11}$$

Based on the values of ϵ_1 and ϵ_2, the attackers can now set the desired (bogus) clock offset at choice and even can select the offset direction. If they delay the `sync` message ($\epsilon_1 > \epsilon_2$), the slave is running ahead in time of the master. In reverse, by delaying the `delay_request` message ($\epsilon_1 < \epsilon_2$), the attackers can set the slave behind. In Sec. III, we introduce different techniques how these delays can be achieved in practice.

D. Attacker Model

For time synchronization algorithms, the attack surface and the attacker model have already been well-studied and summarized in [10]. Especially for delay attacks, two properties are important and often used as reference for further analysis. With the aforementioned security concepts in place, which among others guarantee message authenticity and integrity, it is only possible to create or modify a packet, if the attacker is part of the trusted communication, i.e., has access to the cryptographic keys. In such a case, the attacker is considered to be an *internal* attacker with full capabilities similar to legitimate devices. An *external* attacker, by contrast, has no such access privileges. The attacker's position in the network is the second important property. If relevant PTP messages (e.g., `sync` messages) are routed through the attacker, i.e., direct access to packets is possible, the attacker is considered to be in a *Machine-in-the-Middle (MitM)* position. If the attacker only has access to the network but not to the message itself, e.g., because of sitting in a leaf position, the attacker is considered to be a *packet injector*. As *packet injector*, the attacker can create new packets but has no control over existing packets in the network as these are not routed through.

For our following analysis, we use the presented model to characterize the attack capabilities. The stronger model thereby comprises all capabilities of the weaker form. In particular, an *internal* attacker can perform all actions an *external* attacker can also perform but not vice versa. Similarly, a *MitM* can likewise act as *packet injector*.

III. Attack Implementation

The goal of a delay attack is always the introduction of asymmetric path delay. In this section, we discuss in detail how such attacks can be implemented in real application scenarios and state the required attack capabilities based on the stated attacker model. For each method, slightly different setups are necessary which are illustrated in Fig. 3. In the following, we refer to the appropriate setup and the attack scenario in particular by the assigned letters a to h.

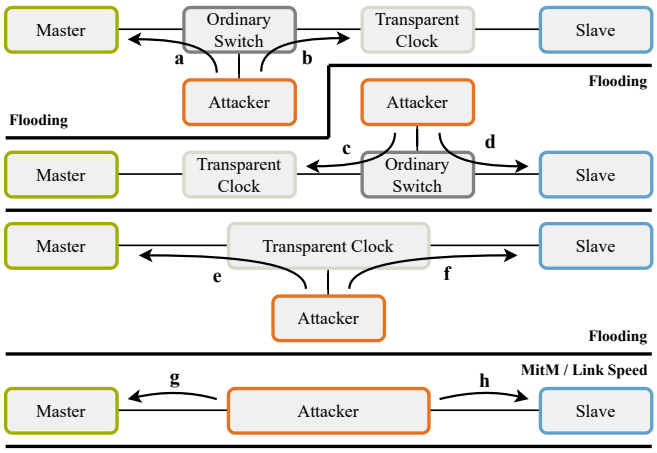


Fig. 3: Overview of the different test setups that we arrange to evaluate the effectiveness of our attack strategies. Each letter thereby refers to an individual attack path that we test.

A. Flooding

The first method to introduce asymmetric path delay is by flooding a specific device with additional network traffic. Usually, flooding is a common technique to perform DoS attacks, but if the network load is carefully regulated, one can still preserve the device's functionality while increasing its processing load, which might affect packet timings. By targeting a particular device, the attacker can control which direction of the path should be congested, i.e., which PTP message is delayed. Attack paths a, c, and e target the master, i.e., the `delay_request` message will be delayed and the slave's clock runs behind. Paths b, d, and f attack the slave which runs ahead in time in this case due to an analogous argumentation. The actual delay is caused by filling packet queues in one direction with dummy messages and, thus, congesting also the PTP messages. To a certain extend, the attacker can even control the delay by adjusting the network load accordingly.

Flooding requires at least one intermediary device like an ordinary switch or a TC between the master and the targeted slave node to inject additional network traffic. Nevertheless, the attacker does not need direct access to the packet nor to be part of the trusted communication. Therefore, the attacker model *external packet injector* does apply. Since the delay is introduced in internal packet queues in the software layer, it results in additional processing delay that only affects the synchronization process if no hardware timestamping is used. Hence, by ensuring that all participating devices provide appropriate hardware support, the flooding attack can be mitigated. In Fig. 3, this means attack paths e and f would not affect the time synchronization.

B. Link Speed

The second method is by changing the link speeds on certain path segments between master and slave. During the negotiation phase at the start of the Ethernet protocol, the two communication partners typically agree on the highest possible link speed that both devices support. This negotiated speed can be different for each individual link in a network. Nevertheless, the path delay for a specific link is still symmetric because

all routed packets are transmitted with the same link speed in both directions. When two consecutive links, however, have different link speeds, we observe the introduction of an asymmetric path delay. The packet needs more time in the direction towards the link with lower speed than in the opposite direction, which resembles a funnel. In one direction, the flow is regulated and congested which affects the packet timings and, thus, also the time synchronization. The attacker has to be positioned along the path, i.e., perform a MitM attack, to force a different link speed at one of the connected links by deliberately advertising a lower link speed than actually supported by the NIC. By forcing the degraded link speed on the path towards the master (g) or towards the slave (h), the attacker can also choose the direction of the clock offset, i.e., whether the slave clock is behind or ahead of the master clock. However, the actual delay cannot be adjusted as the set of possible link speeds is very limited and the caused congestion is hardware-dependent. Since the attacker needs a direct access to the link but is not interested in the exchanged messages, the attacker model *external MitM* does apply.

C. Machine-in-the-Middle

A natural and also reliable method is to sit in a MitM position and implement a custom switching logic that delays specific messages by a certain delay. Hence, the attacker has a fine-grained control over the introduced delay, which makes it a very powerful attack vector. By choosing the appropriate path, the attacker can decide whether the slave clock should run behind (g) or ahead of (h) the master clock. In order to delay a message, the attacker does not require access to the cryptographic keys, but only needs to ensure that the packet is routed through the attack device, i.e., the attacker model is *external MitM*. This also means that especially all cryptographic countermeasures as proposed in prong A and prong B of the PTP standard do not protect against this attack.

D. Correction Field

For completeness, we add another attack method that technically does not introduce any path delay but causes the same effect. For that, the attacker can exploit the CF mechanism for TCs which was actually introduced into the protocol to improve its accuracy. To account for any variable delay in a TC, the packet's residence time is measured and added to the CF variable in the PTP header field. Before the slave is calculating the clock offset, it removes the correction value from the timestamps to obtain a more precise measurement of the link delay. If the attacker has access to this header field and is able to manipulate the value accordingly, the slave clock can be adjusted arbitrarily. With proposed security features such as IPsec and MACsec enabled, the attacker needs, however, access to the cryptographic keys, i.e., internal network access. Hence, this attack is only possible in the *internal MitM* attacker model.

IV. Experiments

After the theoretic attack analysis in the previous sections, we now validate our findings with practical experiments that we perform on our hardware testbed.

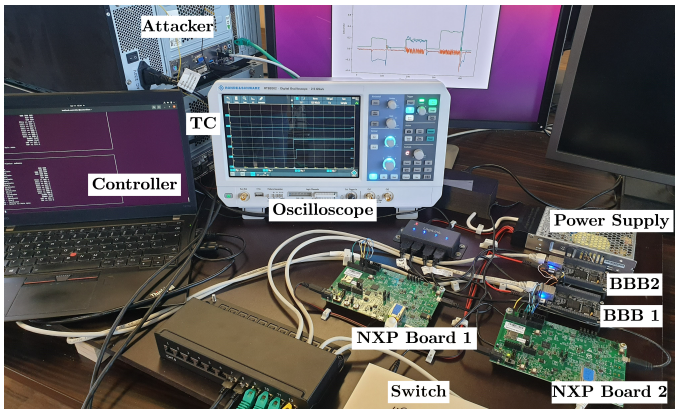


Fig. 4: Realistic testbed to test our time delay attack strategies.

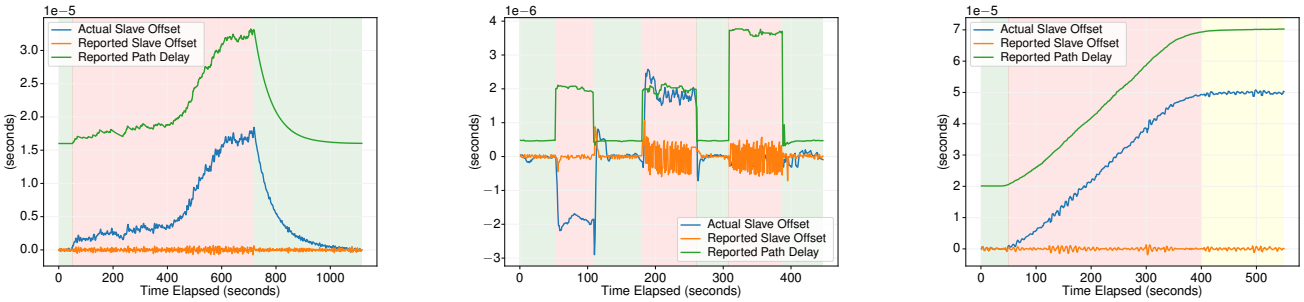
A. Testbed

The testbed consists of two *BeagleBone Black* and two *NXP i.MX RT1050* development boards which are PTP-aware and include one Ethernet port each with support for hardware timestamping. Additionally, both boards provide access to a Pulse-per-Second (PPS) signal that is very helpful for precise performance measurements. Since we use development boards, we also have full control of the software that is running which allows us, for example, to run arbitrary PTP implementations and use the Data Plane Development Kit (DPDK) for performance optimizations. On the BeagleBone boards, we run *LinuxPTP 3.1.1* whereas *PTPd 2.2.2* is installed on the NXP boards to verify the attack applicability independently of the actual PTP implementation. All devices can act as master or slave interchangeably. Furthermore, the testbed comprises two Linux-based PCs that are equipped with *Intel i210* NICs which also provide hardware timestamping. Because the PCs have multiple Ethernet ports, they can operate as intermediary devices, i.e., as TC, BC or attacker device. Also, we include an ordinary switch without special timestamping capabilities to simulate legacy hardware in the network. In order to measure and verify the actual delay between the devices independent of the reported PTP offset that can be attacked, we use a *Rohde&Schwarz RTB2002* oscilloscope to compare the generated PPS signals. Fig. 4 depicts an image of the full setup.

B. Test Cases

The experiments include all depicted attack paths from Fig. 3 and are conducted on the respective setups. All attacks start at $t = 50\,\mathrm{s}$ as indicated by the switch to a red background color in Fig. 5. In green regions, no attack is applied whereas the yellow color marks a change in the attack strategy. The time-to-recover depends on the chosen PTP implementation. Also, we want to highlight that for all experiments PTP is running with IPsec protection enabled.

1) Flooding: For the flooding attack, we gradually increase the applied network load up to $83\,\mathrm{Mbps}$. Fig. 5a depicts the results when targeting a slave device (b). The introduced path delay appears to be approx. piecewise linear with a final value of $18\,\mu\mathrm{s}$ meaning that higher loads have an amplified impact on the synchronization. Interestingly, our measurements show



(a) Flooding Attack (b) Link Speed Attack (c) MitM Attack

Fig. 5: Experimental results of our proposed attack strategies. In the green areas, the system runs under normal conditions, whereas in the red and yellow areas the respective attacks are applied. The results show that the attacker is able to actually delay the slave's clock by several microseconds (actual offset) while the slave keeps the illusion of being synchronized (reported offset). The offset's sign thereby indicates whether the slave clock is ahead of (negative) or behind (positive) the master's clock.

that this method only works in one direction as targeting the master results in a DoS attack independently of the load (a, c).

2) Link Speed: For this attack, we subsequently change the link speed in master direction (g), slave direction (h), and both directions (g, h) from 100 Mb*p*s to 10 Mb*p*s. The change leads to an introduction of approx. 2 µs in the respective direction as illustrated in Fig. 5b. If both links run at reduced speed, the effect cancels out and we only see an increased path delay. This is the expected behavior because the asymmetric delay is caused by the speed difference.

3) Machine-in-the-Middle: For the MitM attack, which is the most flexible attack, we incrementally delay passing packets by 285 ns every second up to a total delay of approx. 10 µs at $t = 400$ s. Afterwards, this level is hold until the end of the experiment as illustrated in Fig. 5c. This test case shows that the attacker is able to precisely control the delay parameter at choice despite having IPsec enabled. The CF attack leads to similar results but requires internal network access to circumvent the IPsec protection.

V. Risk Analysis

As we have seen in the previous sections, there exist several methods to implement time delay attacks on realistic systems. The actual risk, however, depends on multiple factors such as the attacker model that determines the prerequisites for the attacker and the impact on certain security requirements. In the following, we evaluate the proposed attack methods based on the Common Vulnerability Scoring System (CVSS) [11] to estimate the actual risk level. In particular, we focus on the attack complexity, required privileges, and the impact on confidentiality (C), integrity (I), and availability (A). Furthermore, we analyze the actual impact on the time synchronization and state potential mitigation techniques if available. Table I summarizes the results.

The flooding attacks have low complexity and require no further privileges for the attacker to perform them. Although the attacks do not compromise the packet's data integrity, they do affect the timing of it, which can be considered as integrity violation in a broader sense since the packet's timing contains valuable information that gets corrupted. The introduced path delay is not fully deterministic due to the missing direct relationship to the applied network load. Therefore, the respective score is chosen to be *low*. When the flooding attack targets the master, we observe a DoS behavior that renders the time synchronization unavailable. Nevertheless, we only assign *Low* to the availability score to have a further distinction to spoofed clock synchronization which we think is even worse than no synchronization. In contrast, when we flood the path in slave direction, asymmetric delay is introduced which results in a bogus time synchronization and, hence, has a high impact on the availability of correct time. The usage of hardware timestamping (HWTS) on all participating nodes counteracts flooding attacks regarding the distribution of spoofed time. The additional deployment of conventional network monitoring techniques to detect and prevent DoS attacks can further assist in the defense. For all remaining attack techniques, the attacker needs to be in a MitM position which increases the attack complexity. By changing the link speed, the attacker affects the timing and thus the integrity of the packet in a non-deterministic fashion, which results in *low* impact. Despite the possibility to introduce asymmetric path delay, the overall risk is only evaluated to *medium* due to the increased attack complexity. In comparison, MitM attacks provide a fine-grained control over the introduced time delay and, therefore, pose a *high* threat on the timing integrity of PTP messages. With the additional high impact on the time synchronization, the overall risk is stated *high*. While link speed attacks can be detected and counteracted by appropriate network monitoring, there is currently no sufficient countermeasure against MitM delay attacks which renders them even more powerful and dangerous. The CF attack is as such very powerful as well because it impairs all considered security requirements. However, it also requires high privileges, i.e., full communication access in order to modify the PTP message. Hence, the overall risk for the attack results again to *medium* and it can be mitigated by enabling cryptographic countermeasures such as IPsec and MACsec.

VI. Related Work

Security has become an important topic for time synchronization protocols within the last decade and there exist many studies about general attack surfaces and time delay attacks in particular. In [6], Ullmann et al. present a comprehensive

TABLE I: Experimental results and risk analysis based on CVSS metrics. The evaluation shows that time delay attacks cannot be fully mitigated and, thus, still pose a great threat to current systems. The red color highlights attacks with high risk. For the MitM attack in particular, there is also no mitigation available which makes it even more dangerous.

	Type	Attack Path	Attacker Model	Complexity	Privileges	C	I	A	Impact	Risk	Mitigation
a	Flooding	SW-M	Ext-Inj	Low	None	None	Low	Low	DoS	Medium	HWTS / Monitoring
b	Flooding	SW-TC-S	Ext-Inj	Low	None	None	Low	High	Variable	High	HWTS / Monitoring
c	Flooding	SW-TC-M	Ext-Inj	Low	None	None	Low	Low	DoS	Medium	HWTS / Monitoring
d	Flooding	SW-S	Ext-Inj	Low	None	None	Low	High	Variable	High	HWTS / Monitoring
e	Flooding	TC-M	Ext-Inj	Low	None	None	None	None	None	None	-
f	Flooding	TC-S	Ext-Inj	Low	None	None	None	None	None	None	-
g	Link Speed	TC-M	Ext-MitM	High	None	None	Low	High	Fixed	Medium	Monitoring
h	Link Speed	TC-S	Ext-MitM	High	None	None	Low	High	Fixed	Medium	Monitoring
g	MitM	SW-M	Ext-MitM	High	None	None	High	High	Variable	High	None
h	MitM	SW-S	Ext-MitM	High	None	None	High	High	Variable	High	None
g	CF	TC-M	Int-MitM	High	High	High	High	High	Variable	Medium	IPsec / MACsec
h	CF	TC-S	Int-MitM	High	High	High	High	High	Variable	Medium	IPsec / MACsec

mathematical study of delay attacks against PTP, excluding further discussions about practical implementations of such an attack. A similar analysis was performed by Yang et al. with additional Matlab simulations that provide further insights about the effectiveness of the attack [12]. The approach in [13] describes an actual implementation of a delay attack that targets the White Rabbit protocol [14] in Smart Grid systems, which is an extension for PTP. The authors describe and evaluate a delay box that introduces an asymmetric path delay by extending one path direction with additional optical fibers of appropriate length. This attack scenario requires physical access to the network which is a realistic assumption for Smart Grid applications but certainly not applicable to all packet-switched networks. In [8], Annessi et al. discuss different methods to delay messages based on the external MitM attacker model. The attacker can either choose to delay all packets in one direction or specifically filter for PTP which is even possible for encrypted messages due to statistical analysis. Another delay attack experiment is presented in [15], where the authors use a network impairment emulator device to delay PTP messages as part of their comprehensive attack strategy study. Both works focus more on the attack's impact and possible detection or mitigation techniques than on the actual implementation of delay attacks. Han et al. present a physical testbed with further analysis about delay attacks in [16]. The experiments include a MitM as well as a flooding attack. Unfortunately, the testbed is not properly described regarding the used hardware, software or enabled security countermeasures like IPsec and MACsec, which makes a comparison of the results impossible. Furthermore, we focus more on the implementation part of the attacks and provide an additional risk analysis that gives more insights about the actual threat of delay attacks in practical applications.

VII. Conclusion

In this paper, we propose several methods how time delay attacks can be practically implemented and state the required attacker models accordingly. With our open-source testbed, we validate the applicability of all discussed methods and show that current PTP implementations are not sufficiently protected against such attacks. The concluding risk analysis reveals a high demand for further investigations on how to prevent MitM delay attacks since cryptographic countermeasures have proven to be ineffective and the risk for successful attacks is currently very high.

Acknowledgment

This work has received funding from the Bavarian State Ministry of Economic Affairs and Media, Energy and Technology, within the R&D program "Information and Communication Technology", managed by VDI/VDE Innovation + Technik GmbH

References

[1] "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.

[2] J. Tsang and K. Beznosov, *A Security Analysis of the Precise Time Protocol (Short Paper)*. Springer Berlin Heidelberg, 2006, pp. 50–59.

[3] G. Gaderer, A. Treytl, and T. Sauter, "Security aspects for ieee 1588 based clock synchronization protocols," in *Proc. IEEE Int. Workshop Factory Commun. Syst.(WFCS)*. Citeseer, Conference Proceedings.

[4] T. Mizrahi, "Time synchronization security using ipsec and macsec," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2011, pp. 38–43.

[5] B. Moussa, C. Robillard, A. Zugenmaier, M. Kassouf, M. Debbabi, and C. Assi, "Securing the precision time protocol (ptp) against fake timestamps," *IEEE Communications Letters*, vol. 23, no. 2, 2018.

[6] M. Ullmann and M. Vögeler, "Delay attacks—implication on ntp and ptp time synchronization," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2009.

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

[8] R. Annessi, J. Fabini, F. Iglesias, and T. Zseby, "Encryption is futile: Delay attacks on high-precision clock synchronization," *arXiv preprint arXiv:1811.08569*, 2018.

[9] N. M. Freris, S. R. Graham, and P. Kumar, "Fundamental limits on synchronizing clocks over networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, 2010, key Takeaways: - Timestamp data is always better than receiver-receiver causality (without timestamps).

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

[11] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *IEEE Security & Privacy*, vol. 4, no. 6, pp. 85–89, 2006.

[12] Q. Yang, D. An, and W. Yu, "On time desynchronization attack against ieee 1588 protocol in power grid systems," in *2013 IEEE Energytech*. IEEE, 2013, pp. 1–5.

[13] S. Barreto, A. Suresh, and J.-Y. Le Boudec, "Cyber-attack on packet-based time synchronization protocols: The undetectable delay box," in *2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings*. Ieee, 2016, pp. 1–6.

[14] P. Moreira, J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer, "White rabbit: Sub-nanosecond timing distribution over ethernet," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 2009, pp. 1–5.

[15] W. Alghamdi and M. Schukat, "Precision time protocol attack strategies and their resistance to existing security extensions," *Cybersecurity*, vol. 4, no. 1, 2021.

[16] M. Han and P. Crossley, "Vulnerability of ieee 1588 under time synchronization attacks," in *2019 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2019.