MS-PTP: Protecting Network Timing from Byzantine Attacks

Shanghao Shi
Virginia Tech
Arlington, Virginia, USA
shanghaos@vt.edu

Yang Xiao
University of Kentucky
Lexington, Kentucky, USA
xiaoy@uky.edu

Changlai Du
Virginia Tech
Arlington, Virginia, USA
cdu@vt.edu

Md Hasan Shahriar
Virginia Tech
Arlington, Virginia, USA
hshahriar@vt.edu

Ao Li
Washington University in St. Louis
St. Louis, Missouri, USA
ao@wustl.edu

Ning Zhang
Washington University in St. Louis
St. Louis, Missouri, USA
zhang.ning@wustl.edu

Y. Thomas Hou
Virginia Tech
Blacksburg, Virginia, USA
thou@vt.edu

Wenjing Lou
Virginia Tech
Arlington, Virginia, USA
wjlou@vt.edu

ABSTRACT

Time-sensitive applications, such as 5G and IoT, are imposing increasingly stringent security and reliability requirements on network time synchronization. Precision time protocol (PTP) is a de facto solution to achieve high precision time synchronization. It is widely adopted by many industries. Existing efforts in securing the PTP focus on the protection of communication channels, but little attention has been given to the threat of malicious insiders.

In this paper, we first present the security vulnerabilities of PTP and discuss why the current defense mechanisms are unable to counter Byzantine insiders. We demonstrate how a malicious insider can spoof a time source to arbitrarily shift the system time of a victim node on an IoT testbed. We further demonstrate the harmful consequence of the attack on a real Turtlebot3 robotic platform as the robot fails to locate itself and follows a false trajectory. As a countermeasure, we propose multi-source PTP, in short, MS-PTP, a Byzantine-resilient network time synchronization mechanism that relies on time crowdsourcing. MS-PTP changes the current PTP's single source hierarchy to a multi-source client-server architecture, in which PTP clients take responses from multiple time servers and apply a novel secure aggregation scheme to eliminate the effect of malicious responses from unreliable sources. MS-PTP is able to counter f Byzantine failures when the total number of time sources n used by a client satisfies $n \geq 3f + 1$. We provide rigorous proof for its non-parametric accuracy guarantee—achieving bounded error regardless of the Byzantine population. We implemented a prototype of MS-PTP on our IoT testbed and the results show its resilience against Byzantine insiders while maintaining high synchronization accuracy.

WiSec '23, May 29-June 1, 2023, Guildford, United Kingdom

ACM ISBN 978-1-4503-9859-6/23/05.
https://doi.org/10.1145/3558482.3590184

CCS CONCEPTS

• **Security and privacy** → **Network security**; • **Networks** → **Time synchronization protocols**.

KEYWORDS

Byzantine resilience, Network Time Synchronization, Precision Time Protocol (PTP), Service Security and Reliability.

ACM Reference Format:
Shanghao Shi, Yang Xiao, Changlai Du, Md Hasan Shahriar, Ao Li, Ning Zhang, Y. Thomas Hou, and Wenjing Lou. 2023. MS-PTP: Protecting Network Timing from Byzantine Attacks. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23), May 29-June 1, 2023, Guildford, United Kingdom.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3558482.3590184

1 INTRODUCTION

Recently emerged time-sensitive applications, such as autonomous driving and smart grids, usually require a microsecond or sub-microsecond level synchronization between different nodes and failure to achieve these requirements can lead to significant consequences. The Precision Time Protocol (PTP), originally developed by the IEEE 1588 working group [18], is widely regarded as the de facto solution to provide highly precise network synchronization. PTP achieves much higher synchronization accuracy compared to the Network Time Protocol (NTP), the incumbent synchronization service for the Internet. It also offers better flexibility than high-precision GPS-based synchronization, which can only work reliably with outdoor GPS antennas.

Real-world Applications of PTP. With the help of PTP, devices in a local network can be synchronized with sub-microsecond accuracy. Currently, PTP has been documented in many industry standards including the 3GPP 5G standard [5], IEEE TSN standard [4], and IEEE smart grid standard [2], and has been used by various time-sensitive networks such as data center networks [31], and industrial automation networks [35]. In telecommunication, PTP is deployed in the backhaul networks to transfer timing information from accurate time servers in the mobile core networks to

the base station to achieve the 1.5 μs stringent time synchronization requirements of 5G standards [5, 15]. In high-performance data centers such as Meta's data centers, PTP guarantees that data replicas are perfectly synchronized [31] to gain performance improvement as much as 100 times. For CPS, PTP is also employed in autonomous driving vehicles to synchronize ECUs and sensors in the intra-vehicle networks [1].

Gaps in Existing Efforts on Securing PTP. The original PTP [18], however, was designed two decades ago and did not come with any security mechanism against an adverse network environment. It has been shown that earlier versions of PTP are susceptible to various attacks launched by any node in the network, such as message spoofing and replay attacks [7, 13, 24], for its lack of proper authentication mechanism. In response, the latest version of PTP [3] recommends using symmetric key-based authentication mechanisms to counter network adversaries. This mechanism is supported by the latest IETF's PTP key management standard that helps distribute and manage secret keys for PTP nodes [25]. Recent work further argues that symmetric key-based authentication mechanisms are not able to address identity spoofing attacks and need to be replaced by an efficient elliptic-curve and public-key-based cryptography mechanism [24].

While these authentication mechanisms can effectively protect against network packet manipulations using secure communication channels, they cannot defend against malicious or Byzantine insiders. This is because, regardless of how well individual nodes are protected, no system is perfectly secure. In many cases, especially IoT devices or swarm robots, it is quite challenging to ensure all of them are secure, especially since some of them can be physically captured and tampered with by the adversary [9]. Worse yet, the aforementioned cryptographical mechanisms are rarely implemented in popular PTP implementations such as PTPD [23] and linuxPTP [26], introducing more opportunities for attackers.

Analyzing the Impact of Compromised Nodes in Secure Time Synchronization Network. To develop an effective defense against malicious insiders, it is often necessary to have insights into the attack mechanisms. Therefore, the first half of the contribution focuses on the security analysis of the time synchronization network from the perspective of a malicious insider. We found that, even if the communication channel is secure, it remains possible for the adversary to arbitrarily shift the clock of any targeted victim node within the network, using only a single malicious insider. The key idea is to exploit the weakness in the election mechanism for the unique grandmaster (GM) to self-elect as the master to attack the rest of the nodes in the network. To validate our finding, we demonstrated the attack on the two most popular PTP implementations, PTPD [23] and linuxPTP [26], in an IoT testbed. To further show the potentially catastrophic sequence of the attack, we demonstrate the consequence of our attack on a Turtlebot 3 robotic platform. Turtlebot 3 robot relies on synchronized sensor inputs to realize its localization and control functions. When sensors are de-synchronized, the physical world perceptions from different modalities also become desynchronized, leading to errors in the localization and path planning process.

Our Proposed Defense. Our security analysis discovers a key vulnerability that can be exploited by a malicious insider—existing PTP protocols only make use of a single time source. To defend against the attack, we propose MS-PTP, a Byzantine-resilient network time synchronization mechanism to safeguard the dependability and accuracy of PTP timing against a malicious insider who attempts to dis-synchronize clocks of honest clients. MS-PTP leverages redundancy (i.e., multiple time sources) to increase the PTP system's resiliency (i.e., its tolerance to Byzantine failures up to a certain threshold). The redundant time sources provide each client with additional measurements for calculating its clock drift/offset in each synchronization round. However, naively adding time sources (such as taking the average) does necessarily improve the accuracy, since the impact from the malicious input is not bounded. To address this problem, we propose a novel Byzantine-resilient measurement aggregation scheme for the client to obtain a robust estimate of its clock drift/offset, given that f out of the n measurements are potentially Byzantine and $n \geq 3f + 1$. The estimation error of MS-PTP is bounded by $\sqrt{2}$ times the measurement uncertainty of honest PTP sessions.

Evaluation. We implemented a prototype MS-PTP system on our IoT testbed. We validated its resilience against different Byzantine attacks and evaluated its computational efficiency. The results show that MS-PTP is able to retain microsecond level time synchronization accuracy even in the presence of an adaptive attacker with the full knowledge of our defense mechanism. MS-PTP maintains this accuracy when the network size grows to 30, which covers all the typical deployment settings of PTP networks. Moreover, we implemented MS-PTP over network time protocol (NTP), GPS-based time synchronization method, and a mixture of them to complement the PTP-only study. We observed similar Byzantine resiliency performance of MS-PTP for all these protocols, showing the generality of decentralized design across different time synchronization technologies of different scales. Lastly, to understand the theoretical guarantee of the proposed protocol, we've developed proven bounds on the time synchronization error. In summary, this paper makes the following contributions:

- We analyze PTP's vulnerability from an insider adversary perspective. To show the feasibility of the attack, we demonstrate the attack on two popular open software implementations of the PTP protocol in an IoT network testbed. We further demonstrate this attack on a physical indoor robot.
- To thwart the threat of malicious insiders, We propose MS-PTP, a Byzantine-resilient network time synchronization scheme, as an extension to the existing PTP. MS-PTP features a robust measurement aggregation scheme that leverages time crowdsourcing to produce a robust time estimation with a small bounded error.
- We developed a rigorous proof of the correctness of our aggregation mechanism and showed that its accuracy is non-parametric of the population of Byzantine time sources (i.e., fixed error bound). MS-PTP can scale to larger networks and outperforms the current state-of-the-art mechanisms.
- We implemented a proof-of-concept MS-PTP system and evaluated its performance in various network and attack scenarios. The results show that MS-PTP achieves excellent

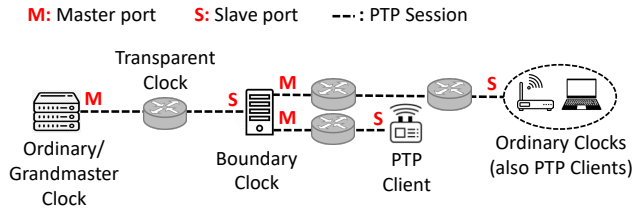


Figure 1: PTP network hierarchy.

synchronization accuracy and is compatible with different time synchronization protocols.

2 PTP: OVERVIEW AND VULNERABILITIES

2.1 PTP Overview

PTP establishes a tree-structured, master-slave network hierarchy, as shown in Fig. 1, to fulfill the time synchronization function. In this architecture, one node, supposedly with the most accurate clock, is elected to serve as the unique time server—the *grandmaster clock* (GM). Other intermediate routers and servers are known as *boundary clocks* (BCs) or *transparent clocks* (TCs). The clients located in the end-leaf nodes are *ordinary clocks* (OCs) in PTP terminology. Under normal operating conditions, the GM periodically delivers timing information to downstream clients through the two-way time transfer (TWTT) mechanism.

Elect the Best Time Source. Upon initialization, the PTP standard specifies all or at least all master candidate devises to continuously broadcast a specific type of PTP message, named the *ANNOUNCE message* on UDP port 320, that contains essential clock accuracy and stability information in its payload. When these messages are received, PTP nodes use the *best master clock algorithm* (BMCA), a clock quality comparison algorithm specified by the PTP standard to decide pairwise master-slave relationship. The master candidate device will stop broadcasting their *ANNOUNCE messages* when they heard from a better clock in the same domain. By doing mutual comparison recursively, the best time source who beats all the other clocks survives to be the grand-master (GM) clock, which becomes the only one that is still broadcasting his *ANNOUNCE messages*. When new nodes join the network, they can obtain current GM's information through reading the GM's *ANNOUNCE messages* and if they have a better clock, can start a new round of this election process.

Two-way Time Transfer. After the time source is elected (i.e. the GM uniquely and stably sends *ANNOUNCE messages* for a certain period), timing information will be periodically transferred from the server to clients following the *two-way time transfer* protocol flow, as fig. 2 has depicted. Within one synchronization round, four accurate hardware timestamps, t_1 to t_4, and two cumulative residence times, c_1 and c_2 are accurately recorded to measure the clock offset θ and clock drift rate δ of the slave node. t_1 refers to the sending time of the SYNC message at the master port and t_2 refers to its reception time at the slave port. Similarly, t_3 is the time that DELAY_REQUEST is sent by the slave port and t_4 refers to the time of its reception at the Master. c_1 and c_2 are measured by down-link and up-link TCs with each one adding its residence time

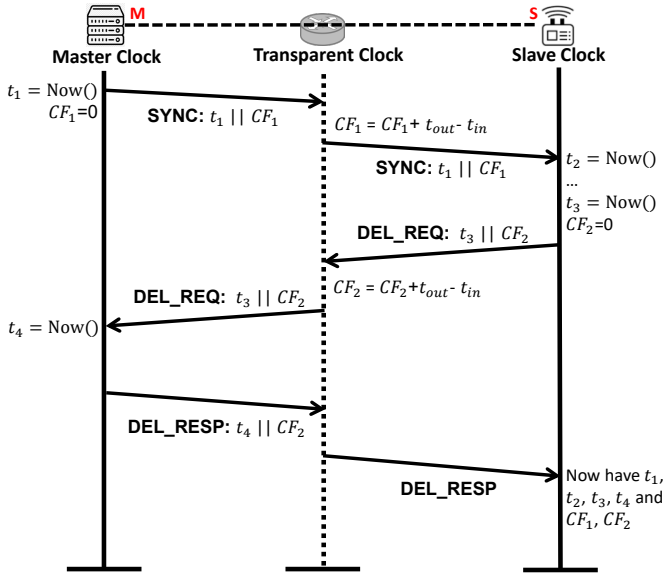


Figure 2: PTP two way time transfer (TWTT) with one transparent clock.

cumulatively to the correction fields of on-the-fly messages. PTP assumes a symmetric path delay δ unless the asymmetry between up-link and down-link path delay is known. The offset θ and path delay δ can be derived as:

$$\theta = \frac{(t_2 - t_1 - c_1) - (t_4 - t_3 - c_2)}{2}$$
$$\delta = \frac{(t_2 - t_1 - c_1) + (t_4 - t_3 - c_2)}{2} \quad (1)$$

For the clock drift rate $\beta - 1$, PTP assumes a constant clock rate β and leverages the offset measured by the current round $\theta(t_{m_1})$ and l^{th} round later $\theta(t_{m_l})$ to get it. l is an adjustable parameter chosen by the clients.

$$\beta = \frac{\theta(t_{m_l}) - \theta(t_{m_1})}{t_{m_l} - t_{m_1}} \quad (2)$$

2.2 PTP Vulnerabilities

The vanilla version of PTP was designed decades ago and has no built-in security mechanism. Recently a revised version of PTP [3] discusses several cryptographic authentication mechanisms, including group key-based direct authentication and TELSA-based delayed authentication [32] to support message authentication. However, these security mechanisms are unable to address malicious insiders [24] and they are not widely adopted and implemented, making PTP almost completely open to all kinds of attacks. In this work, we categorize the attackers into two main classes: network adversaries, who can observe and modify PTP traffics but do not have essential secret keys or credentials to break the cryptography primitives; and malicious insiders, who are compromised legitimate participants of PTP networks.

Network Adversaries. Upon initial design, PTP adopted no security mechanism and can be easily disrupted by simple network-level attacks such as message spoofing, interception, and modification.

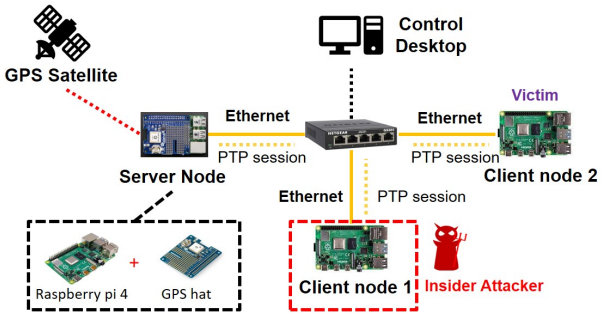


Figure 3: PTP testbed.

[6, 13] introduce several practical fatal network-level time-shifting attacks and validate their feasibility over real-life PTP networks. The symmetric key-based authentication mechanism proposed by the revised standard is also vulnerable to identity spoofing attacks such as rogue grand master attacks [24]. To address all these issues, [24] proposes an elliptic curve-based and public-key-based authentication scheme to establish the authenticity of network clocks. This mechanism can effectively rule out network adversaries by adding small additional overhead to the system.

Malicious Insiders. We consider the malicious insiders to be compromised legitimate participants that possess enough secret keys to bypass the cryptography authentication mechanisms. They may be compromised servers that generate malicious messages from the very beginning or compromised man-in-the-middle (MitM) attackers that modify on-the-fly PTP packets. They cannot be prevented by cryptography methods and pose a great threat to reliable PTP operation. Under the current single-source, tree-structured architecture, it is nearly impossible to detect these attackers cause the downstream clients give full trust to upstream servers and intermediate nodes and there is no way to detect them if they become malicious. In this work, we focus on addressing the most challenging malicious insiders.

3 ATTACK DEMONSTRATION: COMPROMISING PTP NETWORK TIMING

3.1 Experiment Setting

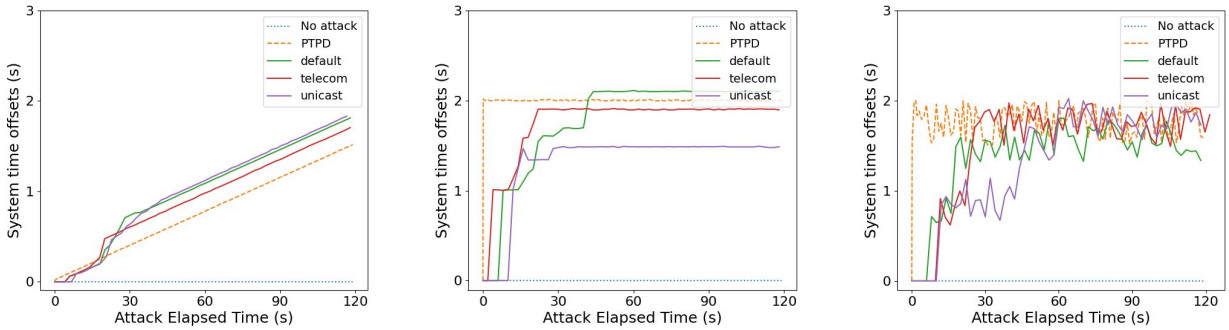
In this section, we demonstrate our experiment about how to shift the time of a victim node on a real IoT testbed. The testbed, as shown in figure 3, contains three nodes including one server node and two client nodes. The server node consists of one Raspberry Pi 4 device and one plug-in GPS hat. The GPS hat can synchronize its clock with the satellites and transfer this nano-second level accurate timing to the Raspberry Pi board, making it a proper time source. The two client nodes are standard Raspberry Pi 4 boards and all three nodes are interconnected via Ethernet. For PTP software, we choose the commonly used implementations – PTPD [23] and LinuxPTP [26] as the victim implementations. PTPD is a classic and well-received PTP daemon on the Linux operating system and supports an older version of PTP standard [18]. LinuxPTP is an implementation of newer PTP standard [3] and supports more profiles. In this work, we launched our attack over the default, unicast, and telecommunication profiles of LinuxPTP.

Attack Threat Model. The attacker is assumed to compromise one node in the time synchronization network, either the server node or a slave node. The attacker shall be able to monitor inbound PTP traffic, get access to the security credentials he received or assigned, and generate and transmit malicious PTP packets. To better understand PTP time-shifting attacks, we adopt a rather weak attack model as the compromised device is a client node because the compromised server and MitM attackers are too strong and it is straightforward and trivial to launch time-shifting attacks with such strong attackers, since these attackers can directly generate malicious packets. We are going to investigate the following four questions in this section: *(a)* Can a compromised client node jeopardize the operation of others in the current PTP networks? *(b)* If so, to what extent can they infect the time of the victim node? *(c)* Can the current defense mechanisms counter these attackers? *(d)* Will our timing attack cause consequences to a real system?

3.2 Insider Time Shifting Attack

We present an insider time-shifting attack launched by a compromised client (the client node 1 in the testbed). The attacker's general attack strategy is first to elect himself to build up a fake network hierarchy and then craft malicious PTP packets to shift the time of a chosen victim node. The attack can be carried out in the following four steps:

- **Phase 1: Clock Information Extraction.** To know the current GM's clock quality, the attacker n_{adv} reads the current transmitting ANNOUNCE messages ANN_h on its UDP port 320 and extracts the current GM's clock quality information q_h in message payloads.
- **Phase 2: Priority Inversion.** The attacker generates a fake clock quality payload q_{adv} that has higher clock quality than the legitimate q_h according to the BMCA-defined clock quality comparison rules. The attacker can increase the clock priority by one or label the clock type as an ultra-high precision atomic clock. The attacker then ensembles a malicious ANNOUNCE message ANN_{adv} according to the authentication mechanism used in the system. The message contains a PTP header, the fake clock quality payload q_{adv}, and a necessary signature or message authentication code used to pass the authentication process. If the system uses digital signature-based method, $ANN_{adv} = pk_{n_i}||q_{adv}||sig_{sk_{n_i}}$. If uses the symmetric key -based method, $ANN_{adv} = q_{adv}||MAC_{sk_{n_i}}$.
- **Phase 3: Injection and Confirmation.** The attacker broadcasts ANN_{adv} periodically through UDP port 320 at the same speed as a normal grand-master clock. The correct transmitting interval can be obtained either through the PTP configuration files or through continuous monitoring of PTP grandmaster's behaviors. The attacker sniffs the incoming messages on this port at the same time and if the attacker fails to receive any incoming ANNOUNCE message for a certain time interval T_{adv}, there is a high probability that nodes in the network have already taken n_{adv} as the new grand-master. In practice, T_{adv} is not a long time period and we usually observed it to be within 10 seconds in our experiment before the attacker seized the grand-master position.



(a) d as a cumulative delay (b) d as a constant delay (c) d as a random delay

Figure 4: Time shifting attack results over PTP implementation PTPD and LinuxPTP. The overall trend shows the effective manipulation of the victim clock offset.

- **Phase 4: Time Shifting.** After the confirmation, the attacker starts the PTP engine to send erroneous information. It follows the normal PTP workflow (i.e. TWTT) but modifies the timestamp field of the SYNC message only to its victim by adding a delay d to t_1. By protocol, the offset measured at the victim device is shifted by $d/2$. This malicious $SYNC_{adv}$ message is also attached with a proper digital signature or message authentication code generated by the shared credentials between the attacker and the victim node.

3.3 Attack Results

We implemented our attack on the testbed with the Python Scapy tool [33], which allows us to sniff and generate arbitrary network packets. We assembled PTP packets according to both the PTP standard and attack methods including crafting proper IP, UDP, and PTP headers, as well as generating PTP payloads and appended signatures/MACs. To evaluate the capability of the attacker, we added the malicious delay d in three different ways—constantly (2s), randomly (mean 1.5s, standard deviation (std) 500ms), and cumulatively (1.8s per 120s). Figure 4 shows the clock offsets of the victim node (client node 2) under different delay-adding methods. We can find that the time (offset) of the victim node was significantly shifted consistently with the way they were manipulated under different PTP implementations and profiles. We observed that the offsets of some LinuxPTP profiles were not immediately shifted when the attacks were launched. There were about 5-10s of delays before they were shifted. But the overall trend of their performance was in line with how they were manipulated. Because the delays we introduced were very large (seconds level) compared to the no-attack clock offsets (microseconds level), the no-attack system offsets became nearly invisible in the figure.

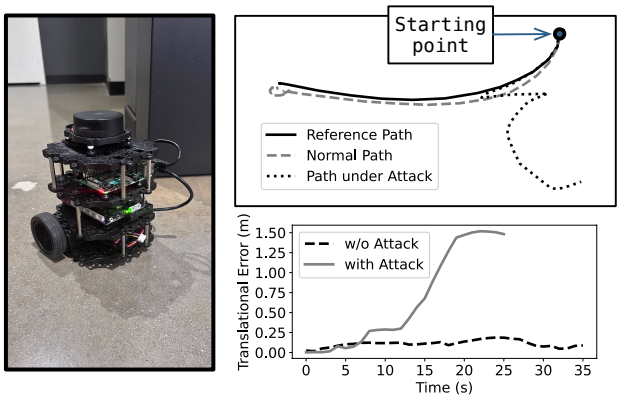
3.4 Case Study: Attack Consequence on a Real Robotic Platform.

To further evaluate the effect of time de-synchronization on real systems, we set up a cloud-based indoor delivery system based on the Amazon RoboMaker [8]. The system consists of an Amazon EC2 cloud server and a Turtlebot 3 robot [37] (as shown in Figure. 5(a)). The robot receives task missions and sensor inputs from the cloud, allowing the robot to navigate in the physical environment according to the commands from the server. The robot takes inputs from multiple sensors such as a camera, inertial measurement unit, and LiDAR. Each sensor is implemented as a separate node in the network and delivery the data via UDP packets. The sensor data is appended with timestamps, which are used to synchronize among different sensors, achieving a consistent sensing result. To emulate the consequence of our PTP timing attack, we added malicious delays to the sensing inputs. This simulates the situation in which inter-vehicle network synchronization is compromised and the sensor inputs are temporally misaligned. Our goal is to investigate whether the timing misalignments, more precisely submillisecond-level or even subsecond-level misalignments, can cause interference with the operation of a real robot.

Figure 5(b) shows the trajectory under normal circumstances as well as the trajectory under attack. It can be observed that the robot navigates well without attack. Conversely, the adversary node can easily trick the desynchronization for up to 2 seconds, leading the robot to fail in localizing itself and navigating to the desired location. As shown in figure 5(b), with a falsely estimated position, the robot arbitrarily moved around the environment and hit the obstacles.

3.5 Attack Analysis and Discussion

With these results, we can answer the questions we have raised before. The compromised client node can shift the time of any victim node arbitrarily by changing the way how it adds delay without any limitation. The reason behind this time-shifting attack is that there lacks a proper clock quality verification mechanism and any node can claim itself to be a high-priority clock source and win the grand-master election process. The current centralized, tree-structured network hierarchy makes things worse for it faces single-point-of-failure and one falsely elected grand master can disrupt the operation of the whole network. While in theory, the system can employ a customized designed public key-based authentication and certificate management system to rule out the spoofing and Sybil attacks [24], they are costly and not used in practice. Furthermore, if we further consider the time servers and MitM nodes are compromised, this defense mechanism cannot help.



(a) Turtlebot3. (b) Control deviations on trajectories.

Figure 5: The consequence of de-synchronization attack on Turtlebot3.

Our defense mechanism defends against insider attacks from a new perspective and can be used in parallel with the current cryptography-based authentication mechanism to further boost PTP's security and reliability. We turn to establish a fault-tolerant and robust time synchronization mechanism, yet some insiders are compromised to exhibit arbitrary behaviors, we assume the majority of the timing devices to be honest. We adopt the fundamental idea of using multiple sources to counter Byzantine sources from the very famous Byzantine fault tolerance state machine replication (BFT-SMR) scheme. This idea is in line with PTP's developing trend – the newest version has already dictated the use of redundant servers to build robust synchronization services [3]. Fortunately, the current PTP leaves room for the simultaneous existence of multiple servers. This can be achieved by properly configuring the PTP software's configuration file. Each PTP client can open up multiple PTP sessions, with each one having its own domain number. By design, different PTP sessions in different domains do not cause interference to others and within each domain, there can be a unique time server. The remaining critical question is how to select a reliable measurement from a set of potentially malicious ones. We will discuss this problem in the next section.

4 MS-PTP: A ROBUST TIME ESTIMATION

4.1 System Model

Countering Byzantine failure is a fundamental problem in distributed networks. A lot of literature has introduced plenty of Byzantine fault-tolerant (BFT) schemes. The most famous BFT schemes may be the *state machine replication* (SMR) solution such as Practical BFT (PBFT) [12] and Tendermint [11]. The fundamental idea of these schemes is to use **redundant nodes** to counter Byzantine minority nodes. We adopt this fundamental idea by designing our Byzantine resilient PTP network using multiple time sources. We also take the threshold assumption from the BFT-SMR schemes, i.e. the number of malicious nodes does not exceed a certain portion of the total population. However, there are key differences in the basic assumptions between classic BFT mechanisms and the problem we are investigating, as we will discuss in the following parts.

Network Model. We assume there are m synchronization sources in the network and each client can connect to n ($n \leq m$) of them to initiate the synchronization procedure. As a result, an individual client is able to receive n independent measurements $(d_1, d_2, \cdots, d_n)$ from n sources in one synchronization round T. Considering the random measurement errors and uncertainty introduced along the communication paths, which is unavoidable in the communication channels, we assume the honest measurements follow the normal distribution of $d_i \sim \mathcal{N}(g, \sigma_i^2)$, where g refers to the correct measurement result under ideal conditions and σ_i^2 refers to the uncertainty level [17]. The key difference between our assumption and the classic BFT-SMR schemes is that the clients in the synchronization problem are not assumed to know the underlying correct measurement g. Therefore, instead of receiving multiple identical correct measurements and a few malicious ones, a time synchronization client is more likely to receive a set of measurements that are different from each other, making it a difficult and non-trivial problem to counter Byzantine measurements among them.

Uncertainty Level. The standard deviation σ_i of an honest time source is considered significantly smaller than the system's required accuracy level r. In fact, the probability that a measurement from an honest source violates the requirement r is $P(|d_i - g| > r) = 1 - \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-r}^{r} e^{-\frac{x^2}{2\sigma_i^2}} \mathrm{d}x$ or in the form of Gaussian error function $P(|d_i - g| > r) = \mathbf{erfc}(\frac{r}{\sqrt{2}\sigma_i})$. This probability is considered as the unreliability rate of the system and is considerably small, otherwise, there will be a non-negligible probability that the requirements are broken. For example, to achieve 5G URLLC's 10^{-7} error rate, r satisfies $r > 5.3\sigma_i$.

MS-PTP Threat Model. We consider insider attackers to exhibit arbitrary behaviors such as delaying and manipulating messages sent to the victim. They may also collude with each other. However, due to the protection from the secure communication channel and honest non-ad-hoc networks, since this scenario provides the best reflection of existing network architecture, one malicious node cannot alter the messages from other nodes in the network. As a consequence, any time synchronization session involving an insider attacker becomes a Byzantine session where a certain number of arbitrary timing measurements are delivered to the clients. The only limitation of the adversary's capability is that they are the minority of the total population and the number of them is smaller than a threshold f. We suppose the compromised measurements do not exceed one-third of the total population.

System Goal. The ultimate goal of our system is to ensure high-precision time synchronization across the PTP network. Given that more than 2/3-majority of measurements are honest ($n \geq 3f + 1$) while others are Byzantium, PTP clients shall be guaranteed to have the final synchronization error smaller than a certain bound that is independent of the number of server population n and Byzantine sessions f.

4.2 MS-PTP-Byzantine Resilient Measurement Aggregation

Based on our assumption, a PTP client receives n measurements in one synchronization round T and f of them are compromised

Algorithm 1 MS-PTP

Input: The number of reachable servers n and the desired number of faults to counter f ($f \leq \lfloor \frac{n}{3} \rfloor$).
Output: System time updated with robust aggregated result o.

```
procedure Periodical Calibration
    function Get measurements
        for i in {1, 2, ··· , n} do
            Get v_i from server n_i.
        end for
        return v_1, v_2, ··· , v_n
    end function
    function Measurements Aggregation
        for i in {1, 2, ··· , n} do
            S(v_i) = Σ_{j∈NM(v_i)} ||v_i − v_j||^2
        end for
        G = (f + 1) arg min_{i∈{1,2,3,···,n}} S(v_i)
        o = 1/(f+1) Σ_{w∈G} w
        return o
    end function
    function Timing Updation
        Update system time with o.
    end function
end procedure
```

and follow arbitrary distributions. We denote these measurements as $(d_1, d_2, \cdots, d_{n-f}, b_1, b_2, \cdots, b_f)$, where $(d_1, d_2, \cdots, d_{n-f})$ refer to the honest measurements and $(b_1, b_2, \cdots, b_f)$ refer to the Byzantine measurements. The general representation of measurements (without knowing its honesty or not) are $v_i (i = 1, 2, \cdots, n)$. We first formally define a robustness criterion for the non-parametric accuracy requirement (in system goal) and introduce our aggregation algorithm $\mathcal{A}^*$ that satisfies this goal.

Definition 1 (Aggregation Robustness). *We define that an aggregation rule $\mathcal{A}$ is r-robust when its output result $o = \mathcal{A}(d_1, d_2, \cdots, d_{n-f}, b_1, b_2, \cdots, b_f)$ satisfies $\|\mathbb{E}[o] - g\| < s < r$, where s is a determined bound independent of f and n. r is the system's synchronization requirement.*

Byzantine-resilient Aggregation Algorithm $\mathcal{A}^*$. We define a score $S(v_i) = \sum_{j \in NM(v_i)} \|v_i - v_j\|^2$, where $NM(v_i)$ includes the $2f$ nearest measurements of v_i. Based on this definition, our aggregation algorithm $\mathcal{A}^*$ can be expressed as:

$$G = (f+1) \underset{i \in \{1,2,3,\cdots,n\}}{\arg\min} S(v_i)$$
$$o = \frac{1}{f+1} \sum_{w \in G} w \tag{3}$$

where $(f+1) \arg\min_{i \in \{1,2,3,\cdots,n\}} S(v_i)$ refers to the set of v_i with the $f+1$ smallest scores; o denotes the output. Algorithm 1 demonstrates the workflow of MS-PTP. MS-PTP takes two parameters including the total number of servers n and the number of failures f as the system input. The clients do not know the exact number of f and usually need to select their desired number of failures to counter. To maximize the fault-tolerant capability, PTP clients usually select $f = \lfloor \frac{n}{3} \rfloor$.

4.3 Theoretical Proofs

Theorem 1 (Correctness). *The Byzantine-resilient aggregation algorithm $\mathcal{A}^*$ is $\sqrt{2}\sigma_{max}$-robust, where $\sigma_{max} = \max\{\sigma_1, ..., \sigma_{n-f}\}$, in which $\sigma_i (i \in \{1, 2, \cdots, n-f\})$ is the standard deviation of honest measurement d_i. To prove this result, the following two lemmas are necessary. The proof of **lemma 1** is trivial and we present the proof of **lemma 2** in the appendix. We focus on the mathematical proof of our main result: theorem 1.*

Lemma 1. *Define $D(d_i) = \sum_{j \neq i} \|d_i - d_j\|^2$ as the sum of the distances between d_i (for $i \in (1, 2, \cdots, n-f)$) and the other honest measurements. It is obvious to have $S(d_i) \leq D(d_i)$ for not all the honest measurements are always within $NM(d_i)$.*

Lemma 2. *There exists at least $f+1$ honest measurements, denoted as $d_k \in \mathcal{B}$, that satisfies $\mathbb{E}[D(d_k)] \leq (2f+2)\sigma^2_{max}$.*

[Proof of Theorem 1] For a measurement v_i, we define the set of honest measurements in its $2f$-nearest neighborhood as $H(v_i)$, with $\|H(v_i)\| = \delta_h(v_i)$ and the set of Byzantine measurements in its $2f$-nearest neighborhood as $B(v_i)$, with $\|B(v_i)\| = \delta_b(v_i)$, Obviously, $\delta_h(v_i) + \delta_b(v_i) = 2f$.

$$\begin{aligned}
\|\mathbb{E}o - g\|^2 &= \|\mathbb{E}[\frac{1}{f+1}\sum_{i=1}^{f+1} w_i] - g\|^2 \\
&= \|\mathbb{E}[\frac{1}{f+1}\sum_{i=1}^{f+1} w_i] - \frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}\sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j\|^2 \\
&= \|\frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}[w_i - \sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j]\|^2 \\
&\overset{\text{(AM-QM Ineq.)}}{\leq} \frac{1}{f+1}\sum_{i=1}^{f+1}\|\mathbb{E}w_i - \sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j\|^2 \\
&\overset{\text{(Jessen's Ineq.)}}{\leq} \frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}\|w_i - \sum_{j\in H(w_i)}\frac{1}{\delta_h(w_i)}d_j\|^2 \\
&= \frac{1}{f+1}\sum_{i=1}^{f+1}\mathbb{E}\|\frac{1}{\delta_h(w_i)}\sum_{j\in H(w_i)}(w_i - d_j)\|^2 \\
&\overset{\text{(AM-QM Ineq.)}}{\leq} \frac{1}{f+1}\sum_{i=1}^{f+1}\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i - d_j)\|^2
\end{aligned} \tag{4}$$

If w_i is an honest measurement, $\mathbb{E}\frac{1}{\delta_h(w_i)}\sum_{j\in H(w_i)}\|(w_i - d_j)\|^2 \leq \frac{\delta_h(w_i)2\sigma^2_{\max}}{\delta_h(w_i)} = 2\sigma^2_{\max}$. Else, if w_i is a Byzantine measurement, $\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i - d_j)\|^2 \leq \frac{1}{\delta_h(w_i)}\mathbb{E}[S(w_i)]$, which by Lemma 2 satisfies $\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i - d_j)\|^2 \leq \frac{2(f+1)\sigma^2_{\max}}{\delta_h(w_i)}$. For a Byzantine measurement, $\delta_h(w_i) \geq f+1$ and $\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i - d_j)\|^2 \leq 2\sigma^2_{max}$.

Combine all the results together, we have:

$$\|\mathbb{E}o - g\|^2 \leq \frac{1}{f+1}\sum_{i=1}^{f+1}\frac{1}{\delta_h(w_i)}\mathbb{E}\sum_{j\in H(w_i)}\|(w_i - d_j)\|^2 \leq \frac{1}{f+1}(f+1)2\sigma_{max}^2 = 2\sigma_{max}^2 \quad (5)$$

In summary, we can prove that $\|\mathbb{E}[o] - g\| \leq \sqrt{2}\sigma_{max}$, which is considered significantly smaller than the synchronization requirements r.

4.4 Analysis

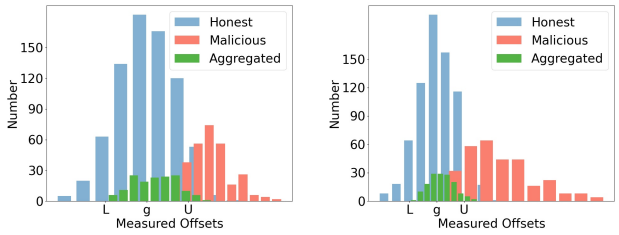
Byzantine Resilience. We have provided a rigorous mathematical proof for the Byzantine resilience of our defense mechanism. The attackers, no matter what kind of behaviors they are doing, are not supposed to break the deterministic error bound. One potential concern the readers raise may be: Can the attackers shift the error bound of the aggregated outputs if they know the defense mechanism and send erroneous measurements accordingly? We investigate the following attack case to provide an intuitive answer.

Case Study: Adaptive Attack. The attackers collude with each other and send malicious measurements near the proved error bound rather than typical random, constant, and cumulative values. The attacker's goal is to gradually break the error bound and shift the victim's time. The attack can be conducted in two steps:

- **Phase 1: Standard Deviation Estimation.** The attackers conclude with each other and thus know all the honest measurements. The attackers estimate the standard deviation of the honest measurements as $std_{adv} = \sqrt{\frac{\sum_{i=1}^{n-f}(d_i - \bar{d})^2}{n-f-1}}$.
- **Phase 2: Adding Malicious Measurements.** The attackers introduce malicious measurements trying to shift the error bound. The malicious measurements can be $\sqrt{2}std_{adv} + \epsilon$, where $\sqrt{2}std_{adv}$ refers to the attacker's estimation of the error bound and ϵ refers to the attacker's desired time shifting direction. Note that ϵ shall be gradually increased from a small value for it will be easily detected and discarded if they are too far away from the error bound.

We took a simulation to investigate and visualize the impact of the adaptive attack and our defense mechanism. The attackers launched the adaptive attack by having ϵ as a gradually increasing value (from 0 to $5\sigma_{max}$) that tries to shift the aggregated result slowly or just having ϵ as a constant small value ($\epsilon = 2 - \sqrt{2}\sigma_{max}$) that tries to shift the distribution of the aggregated results out of the upper or lower error bound. We took our simulation 150 times with the parameters from actual implementations of MS-PTP ($g = 13.95\mu s$ and $\sigma_i = 4.36\mu s$). Figure 6 is our simulation result, where a histogram map is plotted to show the distribution of honest, malicious, and aggregated measurements. We can observe that the attackers failed to do so and only a few points locate out of the error bounds. The simulation result is consistent with our theoretical proof, validating our mathematical results.

Scalability. MS-PTP ensures a deterministic error bound irrelevant to the number of participants. As a result, the theoretical error bound of MS-PTP does not change when the network size becomes larger. Table 1 demonstrates MS-PTP's performance under the adaptive attack in our simulations. We took the state-of-the-art Byzantine fault-tolerant gradient (data) aggregation mechanisms adopted from the federated learning frameworks as a comparison. We took the adaptive attack because it adds malicious measurements near the theoretical bound and can be used to explore the error bound in practice. When there is no attack, the system measures the offset as $g = 13.95\mu s$ and $\sigma_i = 4.36\mu s$. We increased the size of the network from n=4 to n=28 and checked MS-PTP's performance under the attack. We can observe that MS-PTP does not achieve the best performance. But when the network size becomes larger, MS-PTP achieves the best performance among all mechanisms. The output result of MS-PTP remains to be stable and does not increase significantly with larger network size.



(a) ϵ as a constant small value. (b) Gradually increased ϵ.

Figure 6: Adaptive attack over MS-PTP. U refers to the upper bound and L refers to the lower bound.

Complexity. MS-PTP requires multiple (n) redundant servers in the network. As a result, the communication overhead is increased by $n\times$ on the client side. On the server side, MS-PTP does not impose a lot of extra communication overhead for each server that operates in its own domain as a usual one. MS-PTP is a lightweight protocol and does not introduce a lot of computation overhead. The algorithm can be executed within several milliseconds. This is crucial because the algorithm is conducted periodically and a large execution time will pose a significant overhead to the system.

Backward Compatibility. MS-PTP is fully compatible with the current PTP standard. In the cases there are not enough PTP servers such as in the CPS systems, MS-PTP may also resort to alternative external redundancy such as GPS sources and NTP sources. There have already been some cross-protocol synchronization management tools such as Chronyd [34] to help fetch from these available sources.

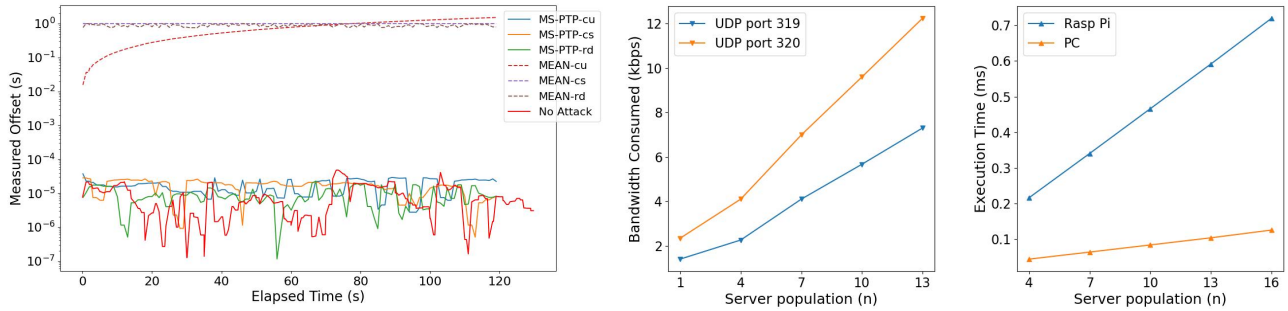
5 EXPERIMENTS AND IMPLEMENTATIONS

5.1 Byzantine Resilience

We implemented MS-PTP on our PTP testbed to evaluate its performance on a real IoT network, where the experimental settings are the same as the attack experiments in Section 3. We added an additional Raspberry Pi to the network and configured all four nodes as server nodes to meet the network redundancy required by MS-PTP. We selected a desktop as a PTP client and configured it to receive measurements from different devices simultaneously. The PTP client captured and took the IP address (in LAN) of all four

Table 1: MS-PTP scalability performance (measured offset in microseconds) under attack.

Scheme	Fault Tolerance	Proved Error Bound	$n = 4$	$n = 10$	$n = 16$	$n = 22$	$n = 28$
Krum [10, 19]	$n \geq 2f + 3$	$\sqrt{2n - 2f + \frac{2f(n-f-2)+2f^2(n-f-1)}{n-2f-2}}\sigma_{max}$	15.407	18.11	20.537	22.478	24.002
Bulyan [19, 21]	$n \geq 4f + 3$	Same as Krum	-	-	-	-	-
Median [19, 39]	$n \geq 2f + 1$	$\sqrt{n - f}\sigma_{max}$	15.880	18.231	19.403	20.167	20.706
Trimmed Mean [39]	$n \geq 2f + 1$	$\sqrt{\frac{2(b+1)(n-f)}{(n-f-b)^2}}\sigma_{max}, 0 \leq b \leq \lfloor \frac{n}{2} \rfloor$	16.205	18.454	19.457	20.153	20.607
Phocas [19, 38]	$n \geq 2f + 1$	$\sqrt{4 + \frac{12(b+1)(n-f)}{(n-f-b)^2}}\sigma_{max}, 0 \leq b \leq \lfloor \frac{n}{2} \rfloor$	18.667	27.413	36.142	44.827	54.447
MS-PTP	$n \geq 3f + 1$	$\sqrt{2}\sigma_{max}$	**15.606**	**16.746**	**17.233**	**17.474**	**17.552**



(a) MS-PTP accuracy performance under attack (cu – cumulative delay, cs – constant delay, rd – random delay)

(b) MS-PTP communication overhead.

(c) MS-PTP computation overhead.

Figure 7: MS-PTP performance on real IoT testbed.

servers and configured its profile to establish four independent sessions with each server. We built up MS-PTP on top of PTP engines with Python code by taking measurements from PTP sessions and aggregating these measurements to obtain a robust timing estimation. This testbed supports hardware validation when the malicious population $f = 1$ and the number of servers $n = 3f + 1 = 4$. We conducted time shifting attack on the compromised node and observed the synchronization performance of the PTP client who employed MS-PTP as a countermeasure.

Fig. 7 shows MS-PTP's performance against time shifting attack, in comparison with the MEAN aggregation method that simply takes the average of measurements as a benchmark. In the experiment, the malicious delay d was added in constant (2s), random (mean 1.5s, standard deviation (std) 500ms), and cumulative (1.8s per 120s) ways inconsistent with the attack settings in section 3. Note that the y-axis is plotted in a logarithm scale to show the results' accuracy level. We can observe that without MS-PTP, the MEAN aggregation's clock offset is shifted by several seconds (10^0). With MS-PTP, the system's measured offsets are maintained at a level of 10 μs (10^{-5}), nearly the same as the results without attack.

5.2 Communication and Computation Overhead

We configured each device to emulate multiple servers in order to testify to the overhead introduced by MS-PTP when the network size becomes larger. For each Raspberry Pi, we launched four PTP sessions and pinned each of them on an independent CPU core respectively. Since Raspberry Pi 4 used in our experiments has a quad-core CPU, we were able to emulate up to 16 PTP sessions/servers in this way. Fig. 7 shows the communication and computation overhead introduced by MS-PTP. For the communication overhead, we monitored the bandwidth consumed by the UDP ports 319 and 320 used by PTP during our experiment on the testbed. We observed tens of kilo bytes-level bandwidth consumption in our experiments. The bandwidth consumption is linearly increasing with respect to n and we take it as a small overhead when n is not large. For the computation overhead, we checked MS-PTP's execution time on one PC and Raspberry Pi. We can also observe a linear increase in the execution time with respect to the server population. MS-PTP's execution time is very small (< 1 ms) on both Raspberry Pi and PC.

5.3 Compatibility

Another important advantage of MS-PTP is that it can not only be used by PTP, but also by the other popular time synchronization mechanisms. We validated MS-PTP's compatibility with NTP and GPS synchronization methods by introducing NTP servers and GPS servers as redundant time sources. We considered the existence of one Byzantine insider, but with either 3 NTP servers or 2 NTP servers plus 1 GPS server serving as the honest redundancy. In these cases, MS-PTP is not only built upon PTP engines but also upon NTP and GPS engines. Fortunately, all of these synchronization mechanisms provide users with convenient interfaces. Any user

with *sudo* priority can get access to them and implement MS-PTP on top of them. Table 2 shows the accuracy performance on our real testbed with these settings when a constant 100 ms delay d is added. We can observe that redundant NTP servers do provide a fault-tolerant guarantee for the synchronization service. The synchronization accuracy level was reduced from $10ms$ level after the attack, to a normal NTP server's $< 1ms$ level. However, because NTP servers' accuracy was worse than PTP servers, the aggregation performance was reduced to NTP accuracy. When a GPS server was added, the aggregation accuracy became better. This indicates that the more accurate servers we are using, the better accuracy MS-PTP can achieve.

Table 2: Measured offset with NTP and GPS servers

Scenarios	Accuracy mean	Accuracy std
Measurements without attack	$13.95\mu s$	$4.36\mu s$
No redundant servers	$37.86ms$	$2.31ms$
3 PTP servers	$12.80\mu s$	$1.32\mu s$
3 NTP servers	$0.974ms$	$0.551ms$
2 NTP, 1 GPS servers	$0.576ms$	$0.578ms$

6 RELATED WORK

6.1 Network Timing Attack and Defense

The original version of NTP and PTP specifies no built-in security mechanism and simple network-level attacks can disrupt them easily. Malhotra et al. [27] introduce the attacks launched with unauthenticated NTP traffic, including on-path time-shifting attacks and off-path DoS attacks. Later in [28], Malhotra et al. demonstrate the security vulnerabilities of the NTP datagram protocol. For PTP, [6, 13] introduce several fatal network-level time-shifting attacks and validate their feasibility over real PTP implementations. Itkin and Wool [24] provide a summary of PTP vulnerabilities and implement the attacks on a popular PTP software PTPD. To counter these attacks, the latest version of PTP [3] recommends using group key-based direct authentication and TELSA-based delayed authentication [32] to support message authentication and identity verification. Furthermore, [24] suggests using an elliptic curve-based public-key signature scheme to establish the authenticity of network clocks. The series of NTP authentication protocols [16, 22, 36] build up authenticated communication channels between servers and clients with appended digital signatures and message authentication codes (MACs). Besides, some stochastic signal processing methods such as the least square-based [29] and Kalman Filter-based [20] estimation methods are introduced to safeguard the timing protocols from malicious noise in the communication paths. To detect malicious timing servers, [30] provides an anomaly detection mechanism with the help of redundant time sources. Deutsch et al. [14] propose using redundant servers and customized data aggregation mechanisms to counter malicious man-in-the-middle (MitM) attackers in NTP networks. However, as we have discussed, the security threats imposed by malicious insiders and how to defend them are largely ignored by the current literature.

6.2 BFT Data Aggregation

Countering against Byzantine measurements among a set of honest ones is an important problem in data mining, especially federated learning. Several Byzantine-resilient gradient aggregation mechanisms have been proposed including Krum [10], Bulyan [21], trimmed-mean [39], median [19] and Phocas [38]. These mechanisms take a similar assumption with our work as the honest measurements are the majority and follow the same distribution, while the Byzantine measurements, although the minority ones, can act arbitrarily. These mechanisms can effectively rule out Byzantine gradients uploaded from compromised clients with proven error bounds. However, they are not designed for the time synchronization problem and their error bounds are not deterministic ones irrelevant to the network population.

7 DISCUSSION

MS-PTP uses network redundancy to counter Byzantine failures. However, some Man-in-the-middle (MitM) attackers are too strong and can not be addressed by MS-PTP. For example, suppose the attacker controls a choke point, where all the communication sessions between the servers and clients pass through. In that case, he can add malicious delays, and no matter how many redundant servers MS-PTP use, still can not find him. Therefore, we recommend that network administrators shall deploy servers in separate locations with as few joint communication paths as possible to avoid such MitM attackers. An individual client should also try to select servers in different geological areas and node-disjoint communication paths from different sources.

8 CONCLUSION

In this paper, we focus on addressing the Byzantine insider attacks in time synchronization systems. We first demonstrated through hardware experiments that the current PTP implementations are susceptible to a practical insider time-shifting attack, in which only one malicious insider is able to bring catastrophe to the time synchronization service. As a countermeasure, we devise a novel Byzantine-resilient aggregation scheme to generate a high-fidelity, error-bounded measurement under the assumption that fewer than one-third of the measurements are Byzantine-influenced. We provide rigorous proof and thorough analysis for the theoretical guarantees that MS-PTP ensures. To evaluate the feasibility and performance of our new mechanism, we implemented a proof-of-concept MS-PTP with a combination of hardware experiments and software implementations in various Byzantine-ridden network scenarios. The result shows that MS-PTP achieves excellent synchronization accuracy for client clocks under different practical attacks. Timing is an important attack vector against the CPS/IoT systems, which have and will be gradually exploited in many time and safety-critical systems. We hope this paper provides a good solution for these time-sensitive systems to set up robust time synchronization services and can motivate more research about the timing vulnerabilities of CPS/IoT systems.

ACKNOWLEDGMENTS

This work was supported in part by the Office of Naval Research under grant N00014-19-1-2621, and the US National Science Foundation under grants 1837519, 1916902, 1916926, 2154929, and 2154930.

REFERENCES

[1] 2015. IEEE Approved Draft Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks - Corrigendum 2: Technical and Editorial Corrections. *IEEE P802.1AS_Cor2/D3.0 July 2015* (2015), 1–11.

[2] 2017. IEEE Standard Profile for Use of IEEE 1588 Precision Time Protocol in Power System Applications. *IEEE Std C37.238-2017 (Revision of IEEE Std C37.238-2011)* (2017), 1–42. https://doi.org/10.1109/IEEESTD.2017.7953616

[3] 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. https://doi.org/10.1109/IEEESTD.2020.9120376

[4] 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. https://doi.org/10.1109/IEEESTD.2020.9121845

[5] 3GPP. 2019. *Study on enhancement of Ultra-Reliable Low-Latency Communication (URLLC) support in the 5G Core network (5GC).* Technical Report. 3GPP TR23.725 V16.2.0 (Release16).

[6] Waleed Alghamdi and Michael Schukat. 2020. Cyber Attacks on Precision Time Protocol Networks—A Case Study. *Electronics* 9, 9 (2020), 1398.

[7] Waleed Alghamdi and Michael Schukat. 2021. Precision time protocol attack strategies and their resistance to existing security extensions. *Cybersecurity* 4, 1 (2021), 1–17.

[8] AmazonRobotics [n. d.]. AWS Robotics. https://aws.amazon.com/robomaker/. Accessed: 2022-05-1.

[9] Manos Antonakakis, Tim April, Michael Bailey, et al. 2017. Understanding the mirai botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*. 1093–1110.

[10] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf

[11] Ethan Buchman. 2016. *Tendermint: Byzantine fault tolerance in the age of blockchains.* Ph. D. Dissertation. University of Guelph.

[12] Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OsDI*, Vol. 99. 173–186.

[13] Casimer DeCusatis, Robert M Lynch, William Kluge, John Houston, Paul A Wojciak, and Steve Guendert. 2019. Impact of cyberattacks on precision time protocol. *IEEE Transactions on Instrumentation and Measurement* 69, 5 (2019), 2172–2181.

[14] Omer Deutsch, Neta Rozen Schiff, Danny Dolev, and Michael Schapira. 2018. Preventing (Network) Time Travel with Chronos.. In *NDSS*.

[15] VP Business Development. 2019. Accurate timing in financial trading. https://www.calnexsol.com/en/timing-and-sync-blog-article-display/1386-accurate-timing-in-financial-trading

[16] Benjamin Dowling, Douglas Stebila, and Greg Zaverucha. 2016. Authenticated network time synchronization. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 823–840.

[17] John C Eidson. 2006. IEEE 1588: an Update on the Standard and Its Application. In *Proceedings of the 38th Annual Precise Time and Time Interval Systems and Applications Meeting*. 193–211.

[18] John C Eidson, Mike Fischer, and Joe White. 2002. IEEE-1588™ Standard for a precision clock synchronization protocol for networked measurement and control systems. In *Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting*. 243–254.

[19] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Louis Alexandre Rouault. 2021. Distributed momentum for byzantine-resilient stochastic gradient descent. In *9th International Conference on Learning Representations (ICLR)*.

[20] Giada Giorgi and Claudio Narduzzi. 2011. Performance analysis of Kalman-filter-based clock synchronization in IEEE 1588 networks. *IEEE transactions on instrumentation and measurement* 60, 8 (2011), 2902–2909.

[21] Rachid Guerraoui, Sébastien Rouault, et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*. PMLR, 3521–3530.

[22] B Haberman, D Mills, and U Delaware. 2010. Network time protocol version 4: Autokey specification. In *RFC 5906*.

[23] IBM. 2019. PTPD Daemon Version 7.2. https://www.ibm.com/docs/en/aix/7.1?topic=p-ptpd-daemon

[24] Eyal Itkin and Avishai Wool. 2017. A security analysis and revised security extension for the precision time protocol. *IEEE Transactions on Dependable and Secure Computing* 17, 1 (2017), 22–34.

[25] M. Langer and R. Bermbach. 2022. NTS4PTP - Key Management System for the Precision Time Protocol Based on the Network Time Security Protocol. https://www.ietf.org/id/draft-langer-ntp-nts-for-ptp-04.html

[26] Linux. 2011. An implementation of the Precision Time Protocol (PTP) according to IEEE standard 1588 for Linux. https://linuxptp.sourceforge.net/

[27] Aanchal Malhotra, Isaac E Cohen, Erik Brakke, and Sharon Goldberg. 2015. Attacking the network time protocol. *Cryptology ePrint Archive* (2015).

[28] Aanchal Malhotra, Matthew Van Gundy, Mayank Varia, Haydn Kennedy, Jonathan Gardner, and Sharon Goldberg. 2017. The security of ntp's datagram protocol. In *International Conference on Financial Cryptography and Data Security*. Springer, 405–423.

[29] Miklós Maróti, Branislav Kusy, Gyula Simon, and Akos Lédeczi. 2004. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*. 39–49.

[30] Bassam Moussa, Marthe Kassouf, Rachid Hadjidj, Mourad Debbabi, and Chadi Assi. 2019. An extension to the precision time protocol (PTP) to enable the detection of cyber attacks. *IEEE Transactions on Industrial Informatics* 16, 1 (2019), 18–27.

[31] Oleg Obleukhov and Ahmad Byagowi. 2022. How Precision Time Protocol is being deployed at Meta. https://engineering.fb.com/2022/11/21/production-engineering/precision-time-protocol-at-meta/

[32] Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. 2002. The TESLA broadcast authentication protocol. *Rsa Cryptobytes* 5, 2 (2002), 2–13.

[33] Relese:2.4.5.dev0. 2022. Scapy: Packet crafting for Python2 and Python3. https://scapy.readthedocs.io/en/latest/

[34] Relese:4.2. 2022. Chronyd: a versatile implementation of the Network Time Protocol (NTP). https://chrony.tuxfamily.org/

[35] Ruxandra Lupas Scheiterer, Chongning Na, Dragan Obradovic, and Günter Steindl. 2009. Synchronization performance of the precision time protocol in industrial automation networks. *IEEE Transactions on Instrumentation and Measurement* 58, 6 (2009), 1849–1857.

[36] Dieter Sibold, Stephen Roettger, and Kristof Teichel. 2016. Network time security. *Work in Progress, draft-ietf-ntp-network-time-security-14* (2016).

[37] Turtlebot3 [n. d.]. Turtlebot3. https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/. Accessed: 2022-011-06.

[38] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2018. Phocas: dimensional byzantine-resilient stochastic gradient descent. *arXiv preprint arXiv:1805.09682* (2018).

[39] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.

APPENDIX

Proof of lemma 2

Without generality, we suppose the honest measurements are within ascending order: $d_1 \leq d_2 \leq, \cdots, \leq d_{n-f}$. First, $d_{f+1} \in \mathcal{B}$, considering that $\mathbb{E}[D(d_{f+1})] = \mathbb{E}\{\sum_{i=1}^{f}[\|d_{f+1} - di\|^2 + \|d_i - d_{2f+2-i}\|^2]\}$ $\leq \mathbb{E}\{\sum_{i=1}^{f}\|d_{2f+2-i} - d_i\|^2\} \leq 2f\sigma_{max}^2$ (Triangular Inequality). Then when $f > 1$, check the expectation of the summation of $D(d_i) + D(d_{2f+2-i})$, $i \in (1, 2\cdots, f)$.

$$\mathbf{E}[D(d_i) + D(d_{2f+2-i})] = \mathbf{E}\{\sum_{j=1}^{i}[\|d_i - d_j\|^2 + \|d_i - d_{2f+2-j}\|^2]\}$$

$$+\mathbf{E}\{\sum_{j=1}^{i}[\|d_{2f+2-i} - d_j\|^2 + \|d_{2f+2-i} - d_{2f+2-j}\|^2]\}$$

$$+\mathbf{E}\{\sum_{j=i+1}^{2f+1-i}\|d_i - d_j\|^2 + \|d_{2f+2-i} - d_j\|^2\} + 2\mathbf{E}\|d_i - d_{2f+2-i}\|^2$$

$$\leq 2 * \mathbf{E}\{\sum_{j=1}^{i}\|d_{2f+2-j} - d_j\|^2 + \mathbf{E}\{\sum_{j=i+1}^{2f+1-i}\|d_{2f+2-i} - d_i\|^2\}$$

$$+2 * 2\sigma_{max}^2 = (2i + 2f - 2i + 2)2\sigma_{max}^2 = (2f + 2)2\sigma_{max}^2$$

Therefore, the expectation of the summation of $D(d_i) + D(d_{2f+2-i})$ is smaller than $(2f+2)2\sigma_{max}^2$, and either $\mathbf{E}[D(d_i)] \leq (2f+2)\sigma_{max}^2$ or $\mathbf{E}[D(d_{2f+2-i})] \leq (2f+2)\sigma_{max}^2$. Because there are f pairs of d_i and d_{2f+2-i}, $i \in (1, 2\cdots, f)$, there exists f other honest measurements $d_k \in \mathcal{B}$.