Protecting Clock Synchronization - Adversary Detection through Network Monitoring

Elena Lisova[†], Marina Gutiérrez[*†], Wilfried Steiner[*], Elisabeth Uhlemann[†],
Johan Akerberg[†], Radu Dobrin[†] and Mats Björkman[†]
[†]School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
{elena.lisova, elisabeth.uhlemann, johan.akerberg, radu.dobrin, mats.bjorkman}@mdh.se
[*]TTTech Computertechnik AG, Vienna, Austria
{marina.gutierrez, wilfried.steiner}@tttech.com

***Abstract*—Today, industrial networks are often used for safety-critical applications with real-time requirements. The architecture of such applications usually has a time-triggered nature that has message scheduling as a core property. Real-time scheduling can be applied only in networks where nodes share the same notion of time, i.e., they are synchronized. Therefore, clock synchronization is one of the fundamental assets of industrial networks with real-time requirements. However, standards for clock synchronization, i.e., IEEE 1588, do not provide the required level of security. This raises the question about clock synchronization protection. In this paper we identify a way to break synchronization based on the IEEE 1588 standard by conducting a man-in-the-middle (MIM) attack followed by a delay attack. MIM attack can be accomplished through e.g., Address Resolution Protocol (ARP) poisoning. Using AVISPA tool we evaluate the potential to perform an ARP poisoning attack. Next, an analysis of the consequences of introducing delays is made, showing both that the attack can, indeed, break clock synchronization and that some design choices, such as a relaxed synchronization condition mode, delay bounding and using knowledge of environmental conditions, can be made to make the network more robust/resilient against these kinds of attacks. Lastly, network monitoring is proposed as a technique to detect anomalies introduced by an adversary performing attacks targeting clock synchronization. The monitoring capabilities are added to the network using a *Configuration Agent*, which, based on data obtained from the network, is able to detect an attack. The main contribution of the paper is a detailed problem description and evaluation of a security vulnerability in IEEE 1588 against delay attacks together with an evaluation of several approaches as possible mitigation techniques for the attack.**

I. Introduction

One of the specific characteristics of industrial networks is a high cost of failure, resulting in money loss, environmental threats or damage to humans. Today, such networks grow extremely fast in complexity and functionality leading to an increasing number of security requirements [1]. Developing a security framework for the whole network implies taking into account application specific features and related systems assets [2]. Clock synchronization is an essential part of all networks with real-time requirements. Basically all industrial networks have real-time requirements, and therefore most messages have deadlines to meet. Consequently, if there is a way to breach clock synchronization, it will disrupt the network functionality, and moreover, it can be applied to a range of different networks regardless of their specific application area [3]. This fact leads to an increased possibility for an adversary to invest resources in such attacks. Also, it is a motivation to investigate possible ways of clock synchronization protection, which includes adversary detection and consequences mitigation.

Most synchronization algorithms, e.g., the IEEE 1588 standard, are vulnerable to delay attacks, as they rely on measuring delays without taking adversary attacks into consideration. A possible way to breach clock synchronization was proposed in [4], namely a combination of an ARP poisoning attack followed by a delay attack. To conduct a delay attack, an adversary first needs to penetrate the network. A so called man-in-the-middle (MIM) attack is one possible way to take control over a communication channel. When performing a MIM attack, the adversary is positioned inside the communication channel between benign participants, e.g., through ARP poisoning. An ARP poisoning attack takes advantage of the vulnerability in the ARP protocol, which is used for translating between IP and MAC addresses. If an adversary can convince a benign network participant N_1 to connect the adversary MAC address to the IP address of another benign node N_2, with whom N_1 wants to communicate, the adversary will get all traffic sent by N_1 to N_2. When the adversary controls the channel, the next step in order to break clock synchronization is to perform a selective delay attack. The combination of these two techniques can be used to breach clock synchronization.

An intrusion detection system (IDS) entails monitoring a network for security purposes. This is not a new approach, and a Network Security Monitor (NSM) system was described already in 1994 [5]. Statistics gathered from a network can be used for many functionalities, such as QoS enhancement, network adaptation for having higher resilience levels, routing optimization, etc. Usually, an adversary penetrating the network changes its behavior in some way. While a passive adversary mainly monitors and analyzes incoming data, the gathered statistics can be used by an active adversary in an upcoming intelligent attack. In case of an active adversary, communication with other network participants is also common. This means that the adversary will introduce new traffic and/or change the traffic pattern. Therefore, if the Monitor, collecting and analyzing the network statistics, can detect a

deviation, the adversary presence can be discovered. It is the first step in network protection, which should be followed by adversary isolation or neutralization and consequences mitigation.

The main contribution of the paper is a detailed problem formulation of clock synchronization vulnerabilities along with evaluation of possible techniques to mitigate consequences of clock synchronization breaking. The investigation of the possibility to detect a malicious adversary targeting clock synchronization breaching in industrial networks using IEEE 1588 for clock synchronization was also conducted. The range of possible attacks is narrowed down to a delay attack performed after a successfully conducted MIM attack. Two scenarios for conducting a MIM attack via ARP poisoning with single and multiple network penetrations are considered. Such choice of scenarios allows to conduct a discussion about the difference in detection and the following adversary localization once they are formally evaluated in AVISPA [6]. ARP poisoning attack is a well known one, but we decided to conduct its formal evaluation, as such approach allows evaluate possible mitigation techniques in future in the same way and compare possible solutions. Further, two different ways of performing a delay attack are simulated in OMNeT++ [7] in order to see how these ways can affect traffic characteristics. The results are also discussed from two points of view: the adversary and the Monitor. Our results show that the likelihood of adversary detection depends on many factors, such as the prior knowledge of the network available to the adversary, the knowledge of the network history available to the Monitor, and, the ability of the network to switch to Relaxed Mode, i.e., allowing additional clock drifts. Environmental conditions are also considered as an additional factor for clock synchronization disturbance and a technique using these conditions for adversary detection is proposed. Finally, another technique that can detect the presence of an adversary, namely delay bounding, is discussed and evaluated.

The remainder of the paper is organized as follows. Sections II presents related works and Section III introduces some background information regarding clock synchronization, in particular the IEEE 1588 standard, and IDS. Next, the system model including models of the adversary and the Monitor are described in Section IV. In Section V, the MIM attack is investigated along with the possible consequences of a delay attack for clock synchronization, whereas Section VI describes the proposed solution for attack detection and countermeasures discussion. Results of the attack evaluation and monitoring simulations are presented in Section VII. Finally Section VIII concludes the paper.

II. Related Works

The initial version of the IEEE 1588 standard does not have any security services. Based on this the authors of [8] show the effects of a delay attack on the IEEE 1588 Precision Time Protocol (PTP). Our work can be considered as an extension of mentioned above paper, as we propose the way of conducting delay attack and widen possible mitigation techniques. In release 2008, Annex K was added to the standard to provide a set of security solutions [9]. However, the amendment provides only a very limited set of services. Annex K provides guidelines for message integrity protection and group source authentication. These security solutions do not help against a delay attack, as in this case the adversary does not need to change the messages or create new ones - it simply delays them. Mizrahi applied game theoretical approach to analyze the delay attack influence on clock synchronization [10]. As a result, the author proposed to use a multiple-path approach to prevent and mitigate delay attacks. However, this approach is not compatible with IEEE 1588.

In [11], network monitoring has been used in the context of synchronized networks. The concept of a Configuration Agent is introduced: an autonomous entity that learns the characteristics of the network through continuous monitoring, with the goal of facilitating the configuration and re-configuration of time-triggered networks. The Configuration Agent is composed of four elements: Monitor, Extractor, Scheduler and Reconfigurator. The duple formed by the Monitor and the Extractor gathers data from the network and distills relevant information from it. In [12], that information is sent to the Scheduler so it can produce a new schedule for the network, with which the network is re-configured. For this paper, we propose to use a generalized view of the same concept, so we replace the Scheduler with a Diagnoser, which after the learning phase performed by the Monitor and the Extractor, will use the distilled information to decide what actions to take. Finally the Reconfigurator carries out those actions.

An example of a network security Monitor running on Ethernet and applied for LANs is presented in [13]. The authors proposed an hierarchical model of data analysis that allows separation of network activities into host-to-host, services, and connections groups. In [14], traffic patterns are proposed to be used for detection of periodic communication of Botnets, subnetworks consisting of infected devices remotely observed by the adversary. These two examples are from different areas and are separated significantly in time, demonstrating that security and monitoring can complement each other in an efficient way.

As mentioned above, applying network monitoring techniques to security issues leads to the development of an IDS. There are two main types of IDS depending on the logic of detection [15]. The first one is a signature-based network IDS. In this approach, there is a set of known attacks together with their corresponding patterns. The IDS is monitoring the system and raises an alarm, when there is a system pattern matching the one from the set. This approach has an obvious limitation, if there is an attack that was not considered at the development phase, it will not be detected. The second group is called heuristic or anomaly-based IDS. With this approach, the system instead knows some standard ways of behavior of the network and search for any anomaly, anything that does not match the standard pattern. The advantage of this method is the possibility to detect a previously unknown attack. On the other hand, if the intruder knows the specific network patterns, the

adversary actions can be masked and indistinguishable from the normal network behavior. The patterns of communication in conjunction with clock synchronization algorithms, which are in the main scope of this paper, are well known. Therefore, in our case we are targeting a hybrid technique, i.e., combining a heuristic and a signature-based method. In this case, the Monitor can be more flexible and have a higher probability of an attack detection.

When evaluating a new solution or mitigation technique, it is essential to have some benchmarks and evaluation criteria. In [16], the authors propose a metric-based approach for IDS evaluation. Logical, architectural and performance metrics were presented as the main groups of criteria. Logical metrics imply such characteristics as cost, maintainability and manageability. Adjustable sensitivity, data pool scalability, data storage, and similar, can be considered as architectural metrics. Finally, error reporting and recovery, induced traffic latency, operational performance impact, and observed false positive and negative ratios, represent performance metrics. The scope of Monitor evaluation used in this paper is to show the impact of the Monitor on the network and the efficiency in attack detection.

III. Background

A. Clock Synchronization

In order to be able to cooperate, nodes of industrial networks have to share the notion of time, i.e., be synchronized. In the ideal case, every node has perfect clock and simply follows the schedule based on its time. In reality, each clock has a natural drift. This drift can be different, mostly depending on the cost of the clock: usually the more expensive the clock is, the more accurate it is. Clock synchronization algorithms are used to assist the nodes with clock correction. Clock drift is a natural characteristic caused by the underlying physical oscillators Therefore, it cannot be completely eliminated, only mitigated, i.e., periodically compensated for. In industrial applications, often only the relative clock differences are important for network correct performance. Consequently, such clocks should be synchronized to each other rather than to an external time reference (e.g., such as GPS).

Clock correction is done periodically as is shown in Fig. 1. The purpose of clock correction is to keep clock time within acceptable boundaries (dashed black lines). The ideal case would be that times provided by the grandmaster clock and the slave clock are always the same (black line with two dots and a dash). But in reality, the slave clock will drift apart from the grandmaster clock (solid green line) and therefore, it needs to be periodically corrected.

B. IEEE 1588 standard

Given two clocks, A and B in a network such as they have been synchronized in a moment in the past, i.e., $t_A = t_B$, in a later moment the time values provided by these clocks will have drifted apart such as:

$$|t_A - t_B| = \sigma. \tag{1}$$

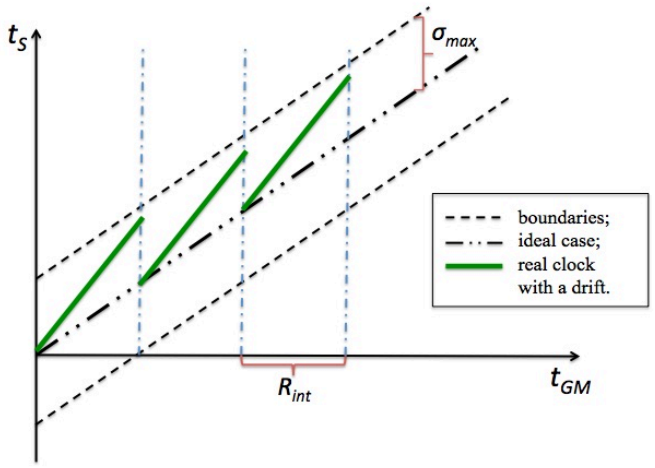

Fig. 1. Periodic correction of slave clock time, t_S according to grandmaster clock time, t_{GM}.

This drift is caused by the non-ideality of the clocks and environmental conditions, e.g., heat affecting the frequency of the oscillators.

To avoid failures, caused by inability of applications to meet their deadlines, clock synchronization protocols are used. Among those, IEEE 1588 is a standard widely used in industrial applications for providing and maintaining clock synchronization [17]. In the IEEE 1588 standard, one of the nodes is chosen as a grandmaster GM, and the rest of the nodes are referred as slaves S_i. Slaves are synchronized to the grandmaster, such that the differences in time values provided by the local clocks in the nodes, as expressed in (1) is bounded. This is expressed in the synchronization condition:

$$|t_{S_i} - t_{GM}| < \sigma_{max} \tag{2}$$

that should be constantly preserved. Here, σ_{max} is a parameter that should be chosen so that the application requirements are satisfied. The minimum value of σ_{max} can be calculated as:

$$\min(\sigma_{max}) = 2 \max(\rho_i) R_{int} \tag{3}$$

where ρ_i are the clock drifts of the clocks in the network and R_{int} is the re-synchronization interval. In Fig. 1, it can be seen how IEEE 1588 works by periodically adjusting the value of the slave clock. The period of these corrections is the re-synchronization interval. The value of the offset used to correct the slave clock is calculated applying the protocol depicted in Fig. 2. The clock synchronization protocol consists of a series of messages being exchanged and time-stamped between the grandmaster and the slave in order to gain enough information to calculate the offset. This process is repeated periodically every re-synchronization interval R_{int}. The message exchange is as follows:

1) At $t = t_1$ the grandmaster sends a synchronization message, (`sync` in Fig. 2) containing t_1 to the slave.

2) At $t = t_2$ the slave receives the `sync` message. Now both t_1 and t_2 are recorded in the slave.
3) At $t = t_3$ the slave sends out the delay request message (`delay_req` in Fig. 2). t_3 is also recorded in the slave.
4) At $t = t_4$ the grandmaster receives the `delay_req` message.
5) At $t > t_4$ the grandmaster sends the delay response message (`delay_resp` in Fig. 2) containing t_4 to the slave. When the slave receives the `delay_resp` message, lastly, t_4 is recorded.

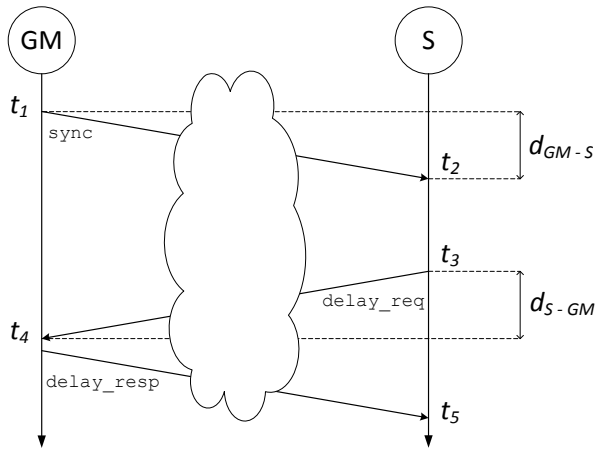

Fig. 2. Clock synchronization protocol.

Finally when all the time-stamps have been collected by the slave, the offset, σ_{meas}, can be calculated according to

$$\begin{aligned} d_{GM\to S} + \sigma_{meas} &= t_2 - t_1, \\ d_{S\to GM} - \sigma_{meas} &= t_4 - t_3, \end{aligned} \tag{4}$$

where $d_{GM\to S}$ and $d_{S\to GM}$ are the transmission delays of a message going from the grandmaster to the slave and from the slave to the grandmaster respectively. Now if we assume that the transmission delay is symmetric, i.e., it is the same in both directions, then $d_{GM\to S} = d_{S\to GM} = d_0$ and the measured offset is:

$$\sigma_{meas} = \frac{1}{2}((t_2 - t_1) - (t_4 - t_3)). \tag{5}$$

Ideally, this value of the measured offset reflects the difference between the grandmaster clock and the slave at the moment when the offset is measured

$$t_S - t_{GM} = \sigma_{meas}. \tag{6}$$

In the IEEE 1588 standard, the equation for offset calculation is more complicated, as there are parameters compensating the propagation delays. For the sake of simplicity they are omitted here, as they do not change the logic of the protocol and do not make any significant difference for conducting a delay attack.

The standard defines three types of clocks, they are *transparent*, *boundary*, and *ordinary*, respectively. A transparent clock performs hardware time-stamping of synchronization messages and updates the corresponding fields in them. A boundary clock has one of its ports in slave mode and gets the time from the grandmaster via this port. It does not update synchronization messages, but can create new ones with the time-stamps according to the information provided by the slave port and send it out. An ordinary clock is a clock without any specific additional functions.

In addition, the standard defines two possible operation modes and two modes for synchronization messages exchange. The following operational modes are possible: end-to-end and peer-to-peer. In the first mode, clocks get information about the delays in links from the exchange of synchronization messages each time they want to make a correction. In the second mode, this exchange of messages is performed for all links regularly and without relation to the clock correction events. Each time a clock wants to correct its time, it has information about all delays with all neighbours. The end-to-end operational mode is suitable for networks where it cannot be guaranteed that all devices in the network support IEEE 1588. The peer-to-peer mode implies that the IEEE 1588 standard is supported by all devices in the network. All above mentioned synchronization message exchanges can be performed in two or in four steps. In the second variant, follow-up messages are used to provide more precise time-stamps.

IV. System model

Considering possible ways of attacking the system and analysing system reaction to the intrusion, it is important to set the limits and assumptions of investigating scenarios. There are many possible ways of interactions between the adversary and the system depending on the assumptions for both. In this paper we consider wired networks, where in synchronization is established and maintained according to IEEE 1588. Using peer-to-peer mode in the network implies that network participants periodically exchange messages to be aware of delays in the channels between them and their neighbours. The networks consist of routers capable of messages time-stamping and nodes, particularly grandmasters and slaves. Routers and nodes have the transparent type of clocks, this means that they perform hardware time-stamping of synchronization messages upon arriving and transmitting, via update of the correction field in the follow-up messages.

Fig. 3 shows the sample topology. The network consist of a grandmaster, a slave, and a set of routers. There are two communication channels between the grandmaster and the slave, namely downlink and uplink.

A. Adversary model and goal

To perform an attack analysis, the adversary model should be specified first. We assume that the adversary has access to and initial knowledge about the network. The adversary knows which node or communication link it is going to attack. The adversary targets clock synchronization breaking, therefore,

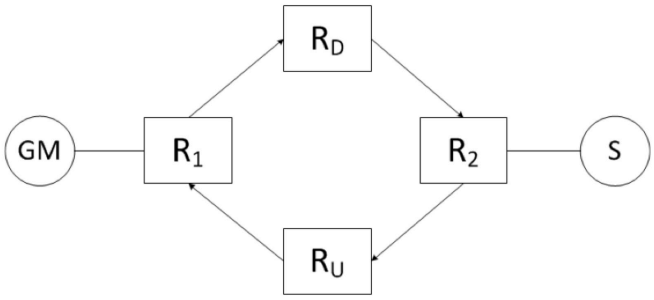

Fig. 3. The Network topology used for the simulations.

it attacks a link which is involved in the synchronization protocol. Except the assumptions mentioned above, adversary choices can be random or based on an analysis of network conducted in advance. We consider a case when the adversary attacks only communication channels, links. In this case, the adversary is capable of receiving, transmitting and delaying messages. At this point, the capability of learning (i.e., possibility to analyse the reaction of the opponent and connect consequences with causes) and behavioural adaptation is not considered. The main adversary goal is system disruption, i.e., the adversary intends to cause system error and propagate it as much as possible, so that it leads to a system failure. Also, the adversary targets to prolong its influence and stay undetected. We assume that the behaviour is rational in pursuing the above-mentioned goals.

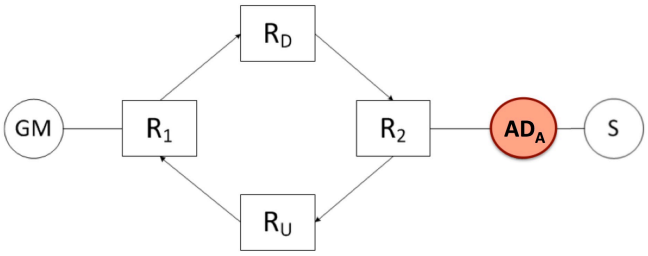

Fig. 4. Scenario A for the network, AD_A - an adversary.

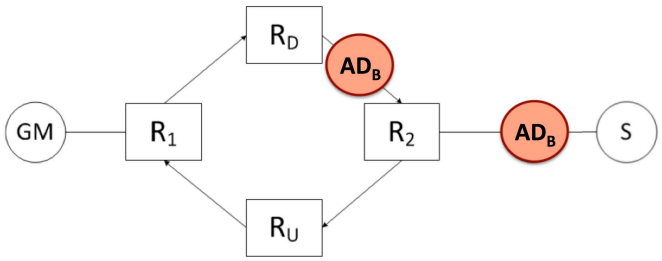

Fig. 5. Scenario B for the network, AD_B - an adversary.

We investigate the case when the adversary uses an ARP poisoning attack to penetrate the network and take control over communications in the targeted channel, i.e., the adversary conducts a man-in-the-middle attack. The next phase for the adversary is to perform a selective delay attack targeting synchronization. As it was specified in the adversary description, the objective of the attack is a link. In this paper we consider two scenarios as is shown in Fig.4-5. Scenario A is a case when one link is under attack. In this case the adversary controls the communication between a router R_2 and a slave S. The second considered case, scenario B, is a consecutive attack on two links. This case is represented by AD_B in Fig.5, the adversary controls communication channels between the router R_D and the router R_2 and between the router R_2 and the slave S. These two links under attack are the parts of one logical channel between the grandmaster GM and the slave S. This scenario implies that the adversary can interfere with the targeted message in different parts of its propagation path (R_D -> R_2 or R_2 -> S). This difference is important for the choice of mitigation techniques. Even though for the slave the results will look exactly the same, during delays analysis it can be more difficult to distinguish the adversary from natural network disturbances. Furthermore, after the adversary detection, it can be more difficult to locate the adversary. Also, such a scenario can be a basis for future modeling of a compromised router. It can be achieved if the possibility to change the contents of the messages is added to the adversary skills set, as then the adversary an actually replace the router from a functional point of view.

B. Configuration Agent Model

The four elements that compose the Configuration Agent that we propose to introduce in the network to detect possible attacks can be seen in Fig. 6. First, the Monitor gathers traffic measurements. Next, the Extractor transforms these traffic measurements in relevant traffic parameters. Furtherafter, by analyzing the traffic parameters, the Diagnoser is able to detect if there are anomalies in the traffic patterns and it determines which are the correct actions to take. Finally, the Reconfigurator changes the configuration of the network to introduce the changes proposed by the Diagnoser.

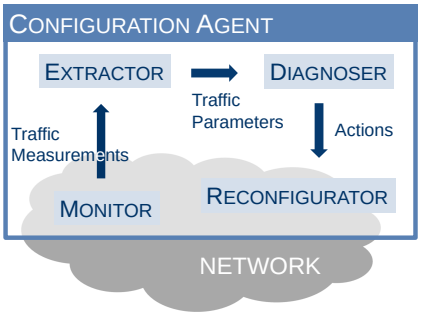

Fig. 6. Configuration Agent overview.

In this paper, we assume that all the functionalities of the Configuration Agent take place locally in every slave in the network. This means that the only information that the Configuration Agent has is the one gathered by the Monitor in a given slave. This approach presents both limitations and advantages. On the one hand it guarantees that the detection and mitigation process is not affected by the same attack that we are trying to detect. On the other hand, having a global view of the network, that allows us to combine the information gathered in every slave, can certainly help with the detection of the attack. That scenario will be explored in future works.

V. Vulnerability analysis

A. ARP poisoning as a method of performing a MIM attack

ARP is a network layer protocol used to define the correlation between MAC and IP addresses for the network participants. When a node N_1 wants to send a message to a node N_2, N_1 knows the IP address of N_2, but to send the message it also needs to know the related MAC address. N_1 first checks its table of IP and associated MAC addresses. If it cannot find the MAC address of N_2 in the table, it sends out a broadcast message requesting the node that has the IP address of N_2 to reply and send back its related MAC address. This communication is completely unprotected and hence vulnerable. Whenever any node receives an ARP reply, it overwrites its table even if it has not sent the request. A MIM attack is an attack when an adversary controls the communication channel between two parties. The parties believe that they communicate with each other directly, but in reality all data exchange is going through the adversary, who possibly can influence the data. An ARP poisoning attack is an attack using the ARP algorithm vulnerability to perform a MIM attack. A malicious adversary can send an ARP response to N_1 pretending to be N_2, and one to N_2 pretending to be N_1. As a result, communication between N_1 and N_2 will go through the adversary. Even though this is a well known attack, it still remains valid and possible, and is used for network penetrations [18].

B. Possible targets of the attack

ARP poisoning re-directs traffic and allows the adversary to control communications between specific sets of MAC addresses. Depending on the adversary goal and the specific application, there are two possible targets of such an attack. Either the adversary can target a concrete communication link, hereafter referred to as scenario *A*, or it can target a specific device in the network, implying that more than one link needs to be compromised, termed scenario *B*. In case of scenario *A*, the adversary controls the communication between the two devices on each side of the link. The adversary can influence both devices in a harmful way if its presence is undetected. Alternatively, a specific device in the network can be chosen as a target. Scenario *B* highlights how an adversary can gain additional advantages, e.g., making its localization more complicated. If the adversary can perform several simultaneous MIM attacks such that it is able to control all incoming and outgoing traffic for one specific device, it simulates the situation of a compromised device through ARP poisoning. Depending on the topology this can have different levels of complexity. The most appealing target for the adversary is most likely a grandmaster clock, as through this clock it can influence all slave clocks connected to it. According to the adversary model, if a clock is compromised, the adversary is capable of creating new synchronization messages, as well as delaying or accelerating messages it forwards. The latter can be achieved by changing the schedule of the clock.In this paper it is assumed that the adversary only performs attacks targeting links, i.e., scenario *A*.

C. Consequences of the delay attack

In the two scenarios *A* and *B* described above, once the first stage of the attack has been performed, i.e., the link has been compromised, the attacker performs the same action in both cases: it introduces a delay in the synchronization messages. In this section, an analysis of the consequences of introducing delays on the time synchronization in the network is shown.

If we use the value of the offset as obtained in (5) to correct the slave clock we have:

$$t_S^{old} \rightarrow t_S^{new} + \sigma_{meas}. \tag{7}$$

Using (7) in (6) we obtain $t_S^{new} - t_{GM} = 0$, making it the best possible value for the offset. This is the value that we would obtain in a attack free situation, therefore henceforth it will be referred as σ_{af}

$$\sigma_{af} = \frac{1}{2}((t_2 - t_1) - (t_4 - t_3)). \tag{8}$$

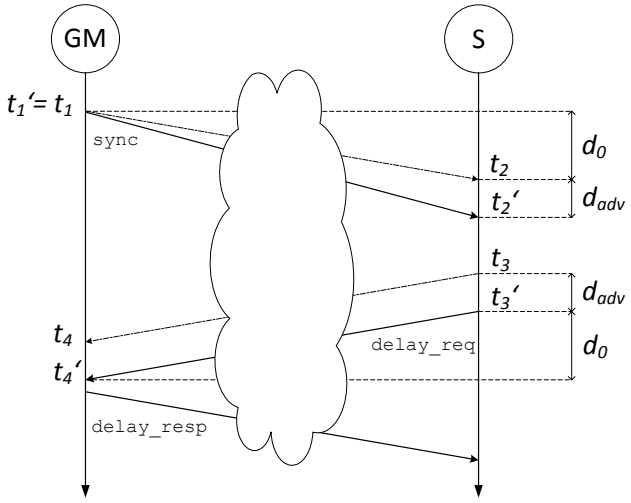

Fig. 7. Clock synchronization protocol under attack.

The value of the offset obtained above, assumes that the transmission delay is symmetric. However, the adversary in the attack that we are considering (Fig. 7) introduces an asymmetric delay, d_{adv} that affects the synchronization messages in the following way:

$$\begin{aligned} d_{GM\to S} =& d_0 + d_{adv} \\ d_{S\to GM} =& d_0. \end{aligned} \tag{9}$$

Now the time-stamps collected by the slave through the synchronization protocol described in Section III-B are:

$$\begin{aligned} t'_1 &= t_1 \\ t'_2 &= t_2 + d_{adv} \\ t'_3 &= t_3 + d_{adv} \\ t'_4 &= t_4 + d_{adv}. \end{aligned} \tag{10}$$

Substituting (10) to (5) and using (8), we obtain the value of the measured offset when there is a delay introduced by the adversary according to:

$$\sigma_{meas} = \sigma_{af} + \frac{1}{2} d_{adv}. \tag{11}$$

If the adversary delays the delay request message instead of the synchronization message, only the sign of d_{adv} in (11) will change. The choice of message to delay does not affect the following reasoning. If σ_{meas} is used to correct the slave clock, then $t_S^{old} \to t_S^{new} + \sigma_{meas}$ and because initially $t_S - t_{GM} = \sigma_{af}$, the difference between the slave clock and the grandmaster is now:

$$t_S - t_{GM} = \sigma_{af} - \sigma_{meas} = \frac{1}{2} d_{adv}. \tag{12}$$

Taking into account that this is the value just after the correction, from where the slave clock will start drifting again and thus for the next re-synchronization interval it will be:

$$t_S - t_{GM} = \frac{1}{2} d_{adv} + \sigma_{meas}. \tag{13}$$

The worst case would be if $|\sigma_{meas}| = \sigma_{max}$, combining this result with the synchronization condition (2), we can conclude that an attack that introduces an asymmetric delay, no matter how small, can break the synchronization. However, this holds true only as long as the chosen σ_{max} for the network is actually the minimum value possible as expressed in (3). In order to make the network more resilient to possible attacks, we can relax this assumption and a longer value can be chosen. Thus, let σ_{rel} be the maximum allowed offset, the synchronization condition would be:

$$|t_{S_i} - t_{GM}| < \sigma_{rel}. \tag{14}$$

Then we obtain a relation between d_{adv}, σ_{max} and σ_{rel} that states that in order to break the time synchronization, an attacker has to introduce a delay twice as long as the difference between σ_{rel} and σ_{max} or, stated differently, a network can tolerate attacks that introduce a delay twice as long as the difference between σ_{rel} and σ_{max} before the time synchronization is broken, thus

$$|\frac{1}{2} d_{adv} + \sigma_{max}| < \sigma_{rel}. \tag{15}$$

According to the defined adversary model, the adversary strives to keep the network penetration unnoticeable. Furthermore, in some cases, the adversary can succeed in keeping the slave ignorant of the synchronization breaking whenever protection techniques are lacking. The best scenario for the adversary is to break the clock synchronization, while letting the slave think that it still is in a synchronized state. If as a result of the attack, the slave still thinks that (14) holds for him, but in reality the offset between the grand master and the slave is bigger than σ_{rel}, the adversary has succeeded. The advantage, from the adversary point of view, of such an outcome is that the system remains oblivious of its failure. This means that the system will not apply any countermeasures to mitigate the consequences and consequently will not be able to return to a safe state.

VI. Potential Solutions and Mitigation Techniques

In this paper we propose the use of a Configuration Agent to detect network penetration via a MIM attack as described above. The idea is that the Configuration Agent will be able to detect the traffic anomalies associated with the attack, and use these to diagnose what is happening.

In addition, we have identified some mitigation techniques that can be used alone or in conjunction with the Configuration Agent to strengthen the IEEE 1588 against delay attacks. These mitigation techniques are not enough by themselves to prevent, protect against or detect an attack, but they can be used to put some boundaries to the damage caused by the attack, thus increasing the resilience of the system.

A. Attack detection

The synchronization messages are sent from the grandmaster with a period equal to the re-synchronization interval, R_{int} and the use of transparent clocks eliminates any possible interference of the rest of the traffic in the network. This means that any variation in R_{int} of the synchronization messages as perceived in the slave could be a hint of something happening in the network.

To detect these anomalies, a Monitor should be placed in the slaves. There the Monitor will be tracking the arrival times of synchronization messages to the slave. The interarrival time between two consecutive messages is:

$$\Delta t_i = t_{i+1} - t_i. \tag{16}$$

Although, the synchronization messages are sent periodically, some variations of the interarrival time should be expected. Nevertheless, an abrupt and sudden change in the interarrival time could be an indication of an attack happening.

Of course, other subtler, smarter attacks are likely not to be detected just by inspection of the interarrival times. For those, we should use some previous knowledge of the network. Here, we will assume that the Configuration Agent has been active in the network for some time before the attack starts and thus we have a history of the arrival times of synchronization messages to the slave. With a set of n values we calculate the average

of the interarrival time, that would be the re-synchronization interval, R_{int}, as perceived by the slave:

$$R_{int} = \frac{\sum_{i=1}^{n-1}(t_{i+1} - t_i)}{n-1} = \frac{t_n - t_1}{n-1}. \quad (17)$$

The main difference between using the interarrival times or the re-synchronization interval as a parameter to detect an attack is that the first is an instantaneous measure that shows right away if something is happening but a smarter attack can go undetected. On the other hand, using the re-synchronization interval it is possible to spot more subtle attacks but some data need to be accumulated before a trend arises.

B. Mitigation techniques

1) Bounding the breach: Recall that the IEEE 1588 clock synchronization protocol is based on the exchange of messages between the grandmaster and the slaves. Concretely, in Fig. 2 it can be seen how the message `sync` is sent at t_1 from the grandmaster to the slave and, as a response, the message `delay_req` is sent from the slave at t_3, arriving to the grandmaster at t_4. Similarly, the slave is waiting the response from the grandmaster, the `delay_resp` message that arrives at t_5. These two request-response relations can be used to prevent that delay attacks take the clock in the slave irreversible away from the grandmaster clock. To do so, we define t_{ret} for the grandmaster and the slaves as the timespan between sending the message and receiving the corresponding response message:

$$\begin{aligned} t_{ret}^{GM} &= t_4 - t_1, \\ t_{ret}^{S} &= t_5 - t_3. \end{aligned} \quad (18)$$

The minimum value for these is the transmission time of the message in the best case, i.e., when it does not suffer any intervention in terms of delay attacks, queuing delays or MAC layer contention. Thus, let n be the number of hops that the message goes through, then the value of t_{ret} can be calculated as

$$\min(t_{ret}) = \frac{messageLength}{dataRate} \times 2n. \quad (19)$$

To calculate the maximum t_{ret}, contention and execution times in the nodes must be taken into account. For the contention we assume that the message can be delayed by at most one message of maximum length in each hop. For the execution time we assume the worst case execution time (t_{WCET}):

$$\begin{aligned} &\max(t_{ret}) = \\ &= \min(t_{ret}) + \frac{maxMessageLength}{dataRate} \times 2n + t_{WCET} \end{aligned} \quad (20)$$

In a small network as the one depicted in Figure 3 with just four hops between the grandmaster and the slave and keeping aside the execution time, the range of t_{ret} is (8, 104) μs (assuming synchronization protocol messages of 126 bytes, a $dataRate$ of 100 Mbps and for the contention using the maximum length for an Ethernet message, 1522 bytes). This means that any value that exceeds that range can imply that the network is under attack. Note however, that with this method only the delay attacks that introduce a delay longer than $\max(t_{ret})$ are detected each time. An attack that introduces a delay of, e.g., 50 μs will not be detected in a situation of low contention even though it is clear from Section VII-B that this is a delay large enough to break synchronization. Hence, this method can not be used alone as a detection mechanism, but only as a mitigation technique to prevent the attack from causing excessive clock drifts.

2) Relaxed Mode: One of the possible network reactions to the detection of an attack is to switch to a relaxed synchronization condition mode. This means that σ_{max} in (2) is increased. This relaxed mode leads to degradation of the network service quality, but may enable fast network recovery. Obviously, the applicability of such an approach depends on the criticality level of the application and the estimated time needed for recovery. It should be mentioned that the ability of the system to switch to the relaxed synchronization condition mode should be considered already during the system development phase.

3) Using environmental conditions: IEEE 1588 targets industrial applications that implies coping with related environmental conditions (e.g., temperature, humidity). These conditions can influence hardware and particularly the clock crystals. To investigate possible consequences for clock synchronization, the message exchange between a grand master *GM* and a slave *S* through a set of routers R_1, ... , R_n is considered, Fig. 8. At each message exchange chain, an error δ, that is caused by hardware time-stamping inaccuracy, is also considered.

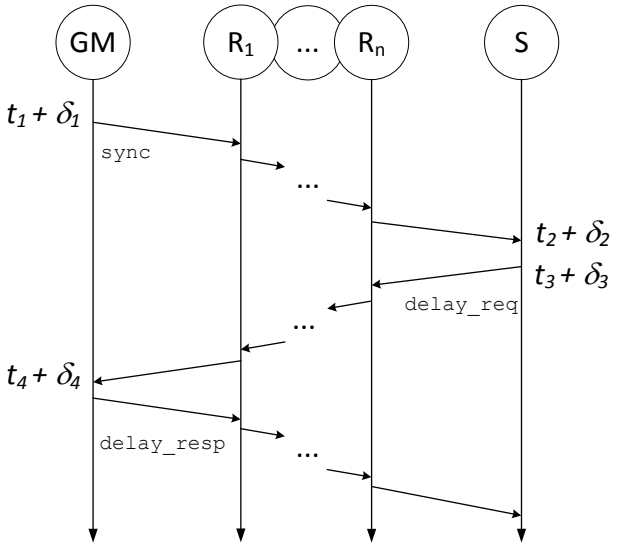

Fig. 8. Synchronization protocol considering additional clock drifts introduced by environmental conditions.

For simplicity first we consider a synchronization message exchange without intermediate nodes, routes, as it depicted in Fig. 7. In this case, in order to take into account additional deviations caused by environmental fluctuations, the following

substitutions are required:

$$\begin{aligned} t_1" &= t_1 + \delta_1 \\ t_2" &= t_2 + \delta_2 \\ t_3" &= t_3 + \delta_3 \\ t_4" &= t_4 + \delta_4. \end{aligned} \tag{21}$$

Then by using (5) we can see that the resulting value of the new offset $\sigma_{meas-envir}$ can be obtained as:

$$\sigma_{meas-envir} = \sigma_{af} + (\delta_2 - \delta_1 + \delta_3 - \delta_4)/2. \tag{22}$$

The values constituting the error δ_i can be grouped according to which node they are produced. Actually, errors made by the same node are similar if we consider events occurring close in time, implying that the events of sending the `sync` and the `delay_req` messages are likely to be subject to the same environmental delay, and conversely, the event of the receiving the sync and the delay requests messages are likely subject to the same delay, according to

$$\begin{aligned} \delta_{GM} &= (\delta_1 + \delta_4)/2, \\ \delta_S &= (\delta_2 + \delta_3)/2. \end{aligned} \tag{23}$$

This yields:

$$\sigma_{meas-envir} = \sigma_{af} + \delta_S - \delta_{GM}. \tag{24}$$

If we add intermediate nodes, routers to the chain of synchronization messages exchange, their corresponding errors will be included in equation (24) twice (once per link they are connected to) but with difference signs. Each intermediate node is an end of the first related link and a beginning of the second related link, i.e., as we consider the difference of these errors for each link, they will have different signs in the final expression. Strictly speaking, when we subtract the two errors associated with the same node, the result is not exactly zero, but it is negligibly small. Therefore, when considering the errors caused by environmental conditions only the grandmaster and the slave errors are significant even if intermediate routes are included in the path. This leads to the conclusion that the most significant influence from the environment occurs when the grand master and the slave are separated far enough, such that they can have different environmental conditions.

This knowledge can also be used for detecting abnormal delays in the communication that cannot be explained by the environmental conditions alone. If nodes have sensors collecting data about main factors influencing clocks crystals, it is viable to calculate possible clock offset between nodes caused by different environmental conditions. If the observed shift is bigger than what can be expected by environmental conditions, this can indicate the presence of an adversary. Having a clock offset bigger than what was estimated can trigger additional checking of, e.g., the links for asymmetry delay detection. Under the assumption of having 5 μs offset with 50 ppm drift, environmental conditions can add 10 ppm to the drift and result in 6 μs offset [19]. This number shows that if the clocks are without temperature compensation, they can affect clock synchronization quite significantly.

C. Countermeasures discussion

When an attack is detected, while the system is in the Relaxed Mode, it should first, try to mitigate existing consequences, i.e., the synchronization breaching and, second, try to prevent the propagation of further consequences. To complete the first goal, related network participants should be informed that there are compromised links. Once the attack is detected, the Monitor could simply indicate between which grandmaster and which slave that there is a breach. The Monitor typically knows the path on which this breach occurred, but it is not known where exactly the adversary is. In the worst case, the whole path from the considered grandmaster to the slave should be eliminated from the clock drift calculations. In Scenario A, the adversary localization can be made by checking, one by one, all the links in the path under the assumption that there is a technique for checking the suspicious link without letting the adversary know about the check. It can be forged synchronization messages, where delaying would directly reveal the presence of the adversary. It is a challenge, as there are many parameters to consider and assumptions to validate. In Scenario B, additional measures should be applied for the adversary localization. In this case, the adversary can act on different links in different order. The possibility of this scenario is as high as the first one, as the adversary does not need any additional techniques for switching from Scenario A to Scenario B. Such switching will bring only benefits to the adversary, as it increases the chances for a successful attack and a longer undiscovered period which in turn means more serious consequences for the network.

VII. Results

A. Attack evaluation

In this subsection, an evaluation of the attack targeting clock synchronization is presented. The attack consists of two phases. The first phase is a MIM attack via ARP poisoning and the second phase is a delay attack. The first phase is evaluated by formal specification of the conducted attack, and the second phase is evaluated by means of logical reasoning in the subsection IV-C. ARP poisoning is not a newly discovered type of the attack and the evaluation made in this paper is an extension of [4], which only considered Scenario A. We conduct a formal evaluation of this attack keeping in mind future work where we need a tool for investigating and comparison mitigation and prevention techniques.

For the formal attack description and evaluation, the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool was used. This tool is used for protocols analysis from a security point of view. It has several possible techniques for evaluation of security properties of the considered protocol, they are the On-the-fly Model Checker (OFMC), the Constraint-Logic-based Attack Searcher (CL-AtSe), the SAT-based Model Checker (SATMC) and the Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP).

The OFMC [20] mode was used in the evaluations, as it allows to include an adversary in the list of network

Type	Identifier
User	R_D, R_2, S, AD_B
Number	IP_{R_D} , IP_{R_2}, IP_S, MAC_{R_D}, MAC_{R_2}, MAC_S, MAC_{AD_B}

TABLE I
IDENTIFIERS

User	Knowledge
R_D	R_2, S, IP_{R_D}, MAC_{R_D}
R_2	R_D, S, IP_{R_2}, MAC_{R_2}
S	R_D, R_2, IP_S, MAC_S
AD_B	R_D, R_2, S, IP_{R_D}, IP_{R_2}, IP_S, MAC_S, MAC_{AD_B}

TABLE II
KNOWLEDGE

participants and by specifying the goal in the correct way, to check the knowledge of all participants in the end of the message exchange.

AVISPA uses High-Level Protocol Specification language (HLPSL) to interact with a user, and there is also a possibility to use the Security Protocol Animator (SPAN) [21] tool to simplify this interaction. Working with SPAN, the user needs to specify several categories of protocol components: identifiers, messages, knowledge and goals. Below, the formalisation of Scenario B is considered. Formal analysis of both scenarios was performed, but as Scenario$B$ can be represented as an extended Scenario A, only the evaluation of Scenario B is described here.

Identifiers. Two types of identifiers were used, they are users and numbers (see Table I). In Scenario B (Fig.5) there are 4 users: R_D, R_2 and S are benign network participants and AD_B is an adversary. IP and MAC addresses of these four users are represented as numbers.

Messages. ARP requests and responses are specified through messages. Intruder AD_B sends a message (ARP request) to R_D, this message contains the IP address of R_D and any other IP address of a benign networks participant. The node R_2 answers to S with a message that contains the MAC address of R_2. After such a manipulation, the adversary obtains the MAC address of R_2. In a similar manner, AD_B obtains the MAC addresses of R_D and S, now AD_B can send messages to all of them. In the next step, AD_B sends to R_D a message (ARP reply) containing the IP addresses of R_D and R_2, plus the MAC address of R_2. After this, for R_D the IP address of R_2 is associated with the MAC address of AD_B. This procedure needs to be repeated for R_2 and S. As a result, R_2 has both the IP addresses R_D and S associated with the MAC address AD_B and S has the IP address R_2 associated with the MAC address AD_B.

Knowledge. Each participant knows its IP and MAC addresses and other benign network participants (see Table II).

Goals. The goal was specified as keeping the MAC address of R_2 secret from R_D and S. If this condition is fulfilled, it means that the attack was performed successfully, as R_D and S possess only the MAC address of AD_B while they think that they communicate with R_2. Therefore, if the tool shows that the secret holds, this means that the adversary wins.

OFMC analysis showed that specified in this way, the protocol is safe for the goal of keeping the MAC address of R_2 secret. This means that the described attack scenario is possible and can be performed. The second step is performing the delay attack. After successfully performing the MIM attack, R_D and R_2 communicate through AD_B, and R_2 and S communicate through AD_B. This means that AD_B can delay selective messages in these two communication channels. As it was shown in Section IV this can lead to clock synchronization breaking.

B. Simulations with OMNeT++

To evaluate the proposed approach, we have created a network simulation using OMNeT++ [7] together with the INET framework [22]. For the concrete modules needed for the simulation of the clock synchronization protocol, the implementation made by Lévesque et al has been used [23]. The goal of these simulations is first to demonstrate that the delay attack can indeed break the clock synchronization. The simulations will also fulfill the role of the Monitor as part of the data that we obtain from them is the same data that a real Monitor would gather in a real network.

Fig. 3 shows the topology of the simulated network. The communication starting in the grand master GM and going to the slave S through the downstream router R_D. Communication starting in the slave goes to the grandmaster through the upstream router R_U. The re-synchronization interval R_{int} is set to 100 ms. We assume that the drift rate of the slave clock is 50 ppm and, therefore, applying (3), $\sigma_{max} = 10$ μs. And we chose $\sigma_{rel} = 20$ μs, thus implying that the system has been designed to work properly with this synchronization accuracy. Without loss of generality, just in order to simplify the explanations of the simulations, we assume that the master has a perfect clock, i.e., it does not drift. However, the slave is, of course, not aware of this fact.

We simulate the effects of an attack that breaks the time synchronization as shown in Section V-C. For that we use different models for the delay: a constant delay and a linearly increasing delay. Once an adversary penetrated the network, it can use different techniques for messages delaying. The goal is to investigate different cases going from the simplest one to more complex and try to analize the differences from the detection point of view.

1) Constant delay, $d_{adv} = 50$ μs: In Fig. 9 the variations of the difference between the slave clock and the grandmaster clock with time is shown. Before the attack the difference was oscillating between 0 and -5 μs, as the result of the clock synchronization protocol performance. After the attack, the difference grows and oscillate between 25 μs and 30 μs. Because these values are bigger than σ_{rel}, we conclude that this attack breaks the clock synchronization. This is the expected result as the value chosen for the delay satisfies (15).

Although Fig. 9 is useful to show how the time synchronization is broken, it can be obtained just in the context of this simulations and not in a real-life situation. To detect the attack

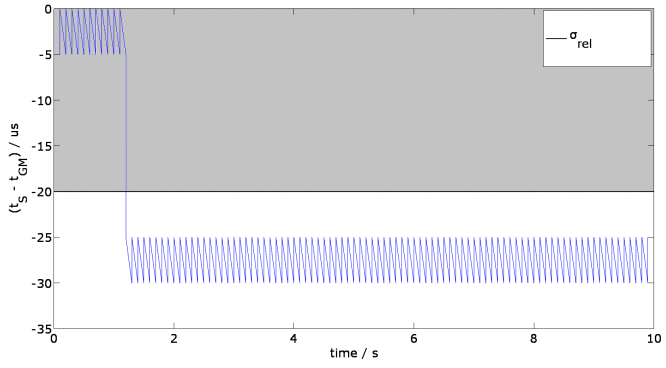

Fig. 9. Time deviation between the grand master clock and the slave clock. Constant delay, $d_{adv} = 50$ μs.

we must restrain the information used to the one available to the slave.

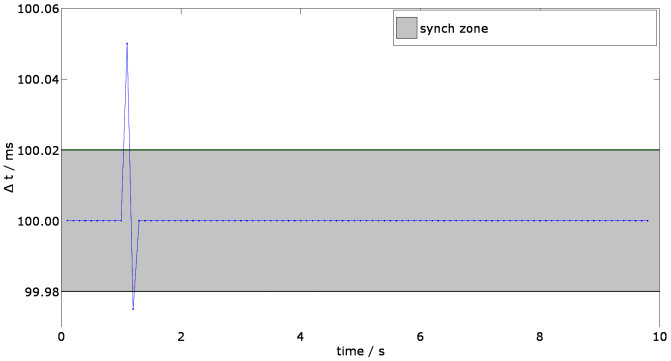

Fig. 10. Interarrival times of sync messages to the slave. Constant delay, $d_{adv} = 50$ μs.

As it was explained in Section VI-A, the Monitor in the slave collects the arrival times of synchronization messages. Fig. 10 shows the interarrival times of synchronization messages to the slave as obtained with (16). Before the attack the interarrival times were constantly equal to the re-synchronization interval. The peek in that figure is the first message affected by the delay. However, all the following messages are also affected by the delay, we can not see it in the figure because the delay is constant, therefore it only shifts the arrival time of the messages, but not the distance between them.

The value of the peak in Fig. 10 ($d_{adv} = 50$ μs) is longer than the maximum possible value ($2\sigma_{max} = 20$ μs), therefore, this attack will be easily detected just by analyzing the interarrival times.

After the attack has been detected, some mitigation techniques can be applied. For example, if the system has been designed to function in a relaxed mode such as $\sigma_{rel} > 30$ μs then the Configuration Agent can carry out this change of mode. Thus, even though the synchronization is deteriorated the system still behave in a predictable manner.

2) Linearly increasing delay: We now simulate a delay that increases linearly with every synchronization message that arrives to the router:

$$d(n) = d_{adv} n, \tag{25}$$

where $d_{adv} = 1$ μs and $n = 1, 2, ...$ for all messages after the attack starts.

In Fig. 11 it can be seen how the initial delay $d(1) = d_{adv}$ is not big enough to break the synchronization, but after enough re-synchronization intervals it does. However, in this case, as compared to the previous one, the attack cannot be detected just by inspecting the interarrival times of synchronization messages to the slave. This can be seen in Fig. 12: the effect of the attack on the interarrival times is so small that the slave might as well confuse it with a drifting in the grandmaster clock. This attack puts the slave in a state in which it is not aware of the fact that it is going out of synchronization when, indeed, it is.

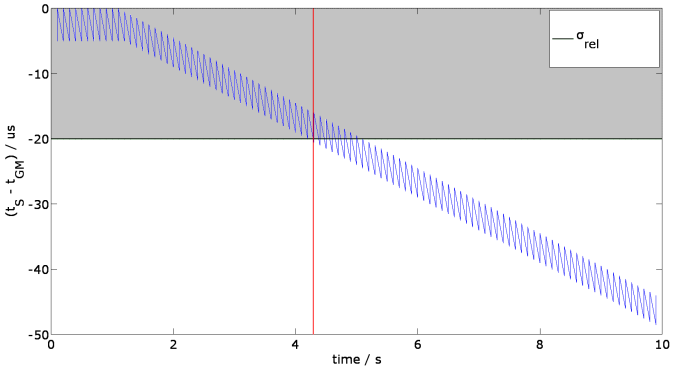

Fig. 11. Time deviation between the grand master clock and the slave clock. Linearly increasing delay.

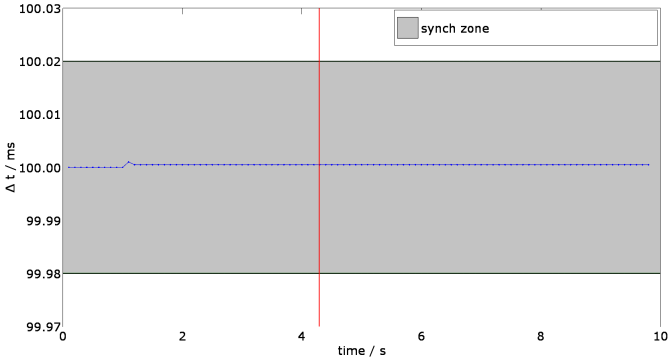

Fig. 12. Interarrival times of sync messages to the slave. Linearly increasing delay.

For this case we conclude that other parameters, different than the interarrival times, should be used to be able to detect the attack. If we want to keep the detection local to the slave, we can assume that the Monitor has been gathering data for

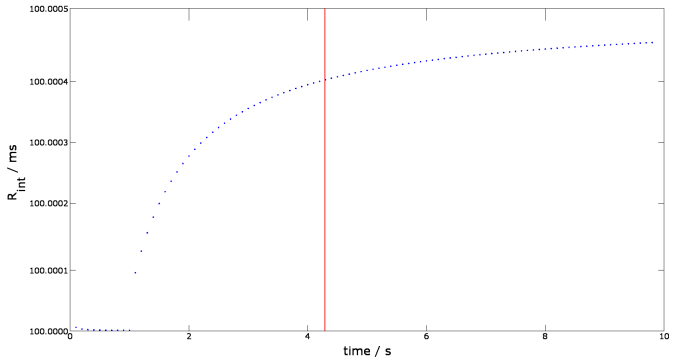

Fig. 13. Re-synchronization interval as obtained in the slave. Linearly increasing delay.

some time before the attack happens and use those values to obtain statistical parameters.

For example, we can examine the variations of the calculated value of the re-synchronization interval obtained using (17). In Fig. 13 it is shown how this value is consistently increasing. Therefore, in order to detect the attack we need to be able to identify this kind of patterns. Because the increase on the value is so small, we could probably not use it a sole criterion to detect an attack, but it can be one of a multi-criteria detection method. A further study could be done comparing the amount of re-synchronization intervals that the attack will need to break the time synchronization with the number of re-synchronization intervals that the Configuration Agent needs to detect the attack.

Independently of the detection capabilities, here we see again how choosing a σ_{rel} longer than σ_{max} gives the network some time to react to the attack even before the synchronization has been broken.

We showed the inner difficulty of attack detection only by means of local monitoring, especially in the case of a smarter attack that introduces a linearly increasing delay. Our proposal is to use mitigation techniques, e.g., like the ones presented in Section VI-B, to cope with attacks that cannot be detected by distributed monitoring in nodes. If we apply the mitigation technique described in Section VI-B.1 to the topology used in the simulations we see that in the worst case scenario (i.e., low contention in the network) this method is able to detect attacks introducing delays longer than 100 μs. This value is way above the minimum delay required to break the synchronization and therefore, the technique cannot be used to prevent time synchronization breaching. Nevertheless, it has two important benefits in the case of the linearly increasing delay. First, it actually allows the slave to detect an attack that it is not possible to detect with the monitoring approach. And secondly, by including the knowledge of this upper limit for the delay in the design, the system can be prepared for this scenario, i.e., has an approach for returning into safe mode.

VIII. Conclusions and Future Work

In this paper, possible strategies of breaching clock synchronization together with techniques on how to detect it were investigated. First, the possibility of conducting the proposed attack breaching clock synchronization was proven by evaluation in AVISPA and through logical reasoning. This conclusion demonstrates the necessity to provide industrial networks with appropriate protection measures. Next a traffic monitoring approach was proposed as a mean of detecting the delay attack. Simulation of the Monitor analysis showed the possibility to detect the delay attack in the case of imposing constant delay. Simulation results also showed that if the system is designed with a relaxed synchronization condition mode, it can help with mitigating the consequences of a delay attack once it has been detected. The localization of the adversary depends on the way of performing the ARP poisoning attack, e.g., in the scenario, when an adversary takes over control of several communication channels, it is more challenging to define which links are compromised. Furthermore, the results also demonstrated that in the case of linearly increasing delay adversary influence can remain undetected. Therefore, more sophisticated detecting techniques are needed to detect such attacks. Algorithms for growing trend detection can then be a possible solution for coping with non-constant delays.

Clock synchronization is an essential part of all networks with real-time requirements. Basically all industrial networks have real-time requirements, and thus if there is a way to breach clock synchronization, the method will disrupt the network functionality, and is applicable for a range of use cases. IEEE 1588 is typically used to provide clock synchronization, but lacks security mechanism. Not even the Annex K, which has been introduced to enhance security, is capable of handling delay attacks such as the ones evaluated in this paper. However, there is nothing in IEEE 1588 which prevents us from using a monitor and thus our proposed solution can be used to enhance the standard.

There is a high potential for future work in this area. We plan to consider more attack scenarios, which include compromised devices and cases with clock acceleration and deceleration. Further, different detection and mitigation strategies, such as distributed monitoring to help locating the adversary together with algorithms for trend detection are to be considered. Furthermore, we want to add learning and adaptation abilities of an adversary and the Monitor to analyze their interactions.

Acknowledgment

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement 607727.

References

[1] D. Dzung, M. Naedele, T. von Hoff, and M. Crevatin, "Security for Industrial Communication Systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, June 2005.

[2] E. Lisova, E. Uhlemann, J. Akerberg, and M. Bojrkman, "Towards secure wireless TTEthernet for industrial process automation applications," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, Sept 2014, pp. 1–4.

[3] E. Lisova, E. Uhlemann, W. Steiner, J. Akerberg, and M. Bjorkman, "A survey of security frameworks suitable for distributed control systems," in *2015 IEEE International Conference on Computing and Network Communications (CoCoNet)*, December 2015.

[4] ——, "Risk evaluation for clock synchronization from arp poisoning attack in industrial applications," in *IEEE International Conference onIndustrial Technology (ICIT2016)*, 2016.

[5] B. Mukherjee, L. Heberlein, and K. Levitt, "Network intrusion detection," *IEEE Networks*, vol. 8, no. 3, pp. 26–41, May 1994.

[6] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heam, O. Kouchnarenko, J. Mantovani, S. Modersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigano, and L. Vigneron, "The avispa tool for the automated validation of internet security protocols and applications," in *Proceedings of the 17th International Conference on Computer Aided Verification*, ser. CAV'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 281–285.

[7] "OMNeT++," http://www.omnetpp.org/, 21 January 2015.

[8] M. Ullmann and M. Vogeler, "Delay attacks - implication on ntp and ptp time synchronization," in *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*, Oct 2009, pp. 1–6.

[9] B. Hirschler and A. Treytl, "Validation and verification of ieee 1588 annex k," in *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2011 International IEEE Symposium on*, Sept 2011, pp. 44–49.

[10] T. Mizrahi, "A game theoretic analysis of delay attacks against time synchronization protocols," in *2012 International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, Sept 2012, pp. 1–6.

[11] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "A configuration agent based on the time-triggered paradigm for real-time networks," in *2015 IEEE World Conference on Factory Communication Systems (WFCS)*, May 2015, pp. 1–4, Best Work-in-Progress Paper Award.

[12] ——, "Learning the parameters of periodic traffic based on network measurements," in *2015 IEEE International Workshop on Measurements & Networking (M&N)*, Oct 2015, pp. 1–6.

[13] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Society Symposium on Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer*, May 1990, pp. 296–304.

[14] M. Eslahi, M. Rohmad, H. Nilsaz, M. Naseri, N. Tahir, and H. Hashim, "Periodicity classification of http traffic to detect http botnets," in *2015 IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)*, April 2015, pp. 119–123.

[15] M. Garuba, C. Liu, and D. Fraites, "Intrusion techniques: Comparative study of network intrusion detection systems," in *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, April 2008, pp. 592–598.

[16] G. Fink, B. Chappell, T. Turner, and K. O'Donoghue, "A metrics-based approach to intrusion detection system evaluation for distributed real-time systems," in *Parallel and Distributed Processing Symposium, Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, April 2002, pp. 8 pp–.

[17] "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, July 2008.

[18] B. Kang, P. Maynard, K. McLaughlin, S. Sezer, F. Andren, C. Seitl, F. Kupzog, and T. Strasser, "Investigating cyber-physical attacks against iec 61850 photovoltaic inverter installations," in *Emerging Technologies Factory Automation (ETFA), 2015 IEEE 20th Conference on*, Sept 2015, pp. 1–8.

[19] TCXO, Temperature Compensated Crystal Oscillator. [Online]. Available: http://www.radio-electronics.com/info/data/crystals/tcxo.php

[20] D. Basin, S. Modersheim, and L. Vigano, "Ofmc: A symbolic model-checker for security protocols," 2004.

[21] Y. Glouche, T. Genet, O. Heen, and O. Courtay, "A security protocol animator tool for avispa," in *In ARTIST-2 workshop*, 2006.

[22] "INET Framework," https://inet.omnetpp.org/, 21 June 2015.

[23] M. Levesque and D. Tipper, "ptp++: A Precision Time Protocol Simulation Model for OMNeT++ / INET," *CoRR*, vol. abs/1509.03169, 2015. [Online]. Available: http://arxiv.org/abs/1509.03169