Monitoring of Clock Synchronization in Cyber-Physical Systems: A Sensitivity Analysis

Elena Lisova, Elisabeth Uhlemann, Johan Åkerberg, Mats Björkman
Mälardalen University, Västerås, Sweden
{elena.lisova, elisabeth.uhlemann, johan.akerberg, mats.bjorkman}@mdh.se

***Abstract*—Clock synchronization is a core asset to protect when securing cyber-physical systems with a time-triggered architecture. One of the most challenging attacks to protect against is a delay attack, where an adversary delays one of the synchronization messages, making node offset calculations incorrect for keeping clocks synchronized. One way to detect a breach of clock synchronization is by monitoring the offsets calculated in a node according to the clock synchronization algorithm. The analysis in this work assumes that the distributed nodes need to share the same notion of time and for this reason, uses the IEEE 1588 standard. Using this approach, a monitor needs to make a decision about if and when a node is under attack, in which case rules and methods for decision making should be put in place. There are many aspects to consider when setting thresholds for the monitored values in order to make such a decision. In this work we conduct an analysis of monitor indicators and an investigation of their applicability. Further, we identify dependencies within the proposed monitoring approach and conduct a sensitivity analysis of the parameters needed to make a decision about a system being under attack. The analysis outcomes allow to identify important parameters to consider while thresholding indicators and enables a greater generality in their applicability.**

***Index Terms*—clock synchronization; offset monitoring; indicators; sensitivity analysis; IEEE 1588**

I. Introduction

Having a monitor in the network of embedded systems has several benefits; it can increase the level of safety in the system, e.g., by detecting component failures, and/or the level of security in the system, e.g., by detecting an attack being deployed. A monitor used for safety reasons detects system states leading to possible hazards, with the goal to prevent or at least mitigate the consequences. A monitor used for security purposes detects malicious actions in the network with the goal to suppress adversary actions and their consequences. Thus, the actions required from a monitor to cover both domains are similar and thus a monitor can be used to increase the dependability for safety-critical embedded systems.

For many embedded systems, it is critically important to detect, in time, a breach of system safety or security to be able to prevent or mitigate the consequences. In the case of safety, this is typically done by switching the system into a safe state and taking necessary precautions. In the case of security, a quarantine state is often used. To make a decision whether a system has malfunctioning components or behavior, various indicators can be monitored and corresponding thresholds can be used. However, as there are also natural disturbances in networks, it is not always straight forward to distinguish between a benign and a malign behavior. Therefore, several network states include a situation when there is only a suspicion about an anomaly or an attack being deployed, but the confidence level is low. In this case, the quarantine state can be used, in which additional indicators are monitored in order to increase the probability of anomaly detection and capture different aspects of network behavior.

This work uses a monitor approach with three network states, normal, quarantine and anomaly detected, as introduced in [1]. Note that the analysis presented in this work is made under the assumption of IEEE 1588 being used to establish and maintain clock synchronization, similar to [1]. The monitor switches states based on inputs, called indicators. An indicator is a value obtained by monitoring, which usually has two thresholds, a low and a high. Although monitoring can be used for many purposes, the work herein considers its configuration related to security and, in particular, targeting protection of clock synchronization for cyber-physical systems (CPS). An embedded system is a type of CPS, and as such, timing is an essential asset as these systems interact with their surrounding environment. To this end, a set of indicators for monitoring clock-synchronization and a way to threshold them were proposed in [2]. However, although a general way to threshold the indicators was considered, it is not straightforward how to generalize the outcome of the analysis and thereby estimate the efficiency of each indicator. There are two main sources of uncertainties. Firstly, each particular application has its own parameters, functionality and precision, which dictates the load for each node, and thereby, defines, e.g., for how long an adversary needs to perform an attack to cause consequences or how many components that needs to be faulty in order to affect safety. Secondly, an adversary model was used as an input for some indicators in [2], and thus, the considered adversary strategies and attack space limit the approach and affects the outcome of the analysis. Therefore, a sensitivity analysis is needed to determine how different inputs are affecting the thresholds for the indicators to be able to generalize the results. The contribution of this work is an analysis of monitor indicators and an investigation of their applicability. Within the analysis, the dependencies of the thresholds of the indicators with respect to adversary model and use case are investigated. To identify connections to a use case specification, an example of a CPS is considered. The outcome of the analysis allows to generalize the applicability of different indicators and identify limitations of the approach, i.e., providing grounds for further

monitor enhancements.

Section 2 introduces the necessary background about the use case: CPS and clock synchronization, whereas Section 3 presents the adversary and monitor models, together with the use case and its corresponding indicators. The general sensitivity analysis of the monitor indicators is conducted in Section 4, whereas Section 5 presents a sensitivity analysis on a particular example of a CPS. A discussion about the approach and its applicability is presented in Section 6 and finally, Section 7 concludes the paper.

II. Monitoring Use Case

A. Cyber-Physical Systems

In CPS, computations and physical processes are combined in order to provide new services enabled by the control loops between these two instances. CPS include a wide range of applications: semi-autonomous vehicles, advanced control systems, distributed robotics, health monitoring systems, etc. Such systems open up new possibilities, e.g., for the next generation of transportation systems, they can enable vehicle health monitoring, as well as its (semi)-autonomous functionalities [3]. Communications in CPS is an example of networks with real-time requirements, implying that they have time as a functional requirement. CPS with a time-triggered architecture require nodes within them to follow a schedule for message exchange and in order to maintain a common notion of time, clock synchronization is the foundation on which such an architecture is built upon. Therefore, in time-triggered CPS clock synchronization can be considered as one of the main system assets, since if it is breached, the whole network is disrupted. The price of its breach is especially high for safety-critical applications, where a system failure could have catastrophic consequences. Thus, for such systems, clock synchronization protection should be addressed.

B. Clock Synchronization

The notion of time is important for networks supporting real-time requirements. Each node usually has its own clock, but no matter how costly, each clock has a natural drift and with time, the difference between different clocks can be significant. Clock synchronization is typically established and maintained by a clock synchronization algorithm. There are several widely used algorithms such as Precision Time Protocol (PTP) and Network Time Protocol (NTP). PTP is a part of the IEEE 1588 standard [4] commonly used in industrial applications. The standard has some optional security guidelines in Annex K, however, these are not sufficient considering the criticality of the asset [5], e.g., since the protocol cannot cope with a delay attack [6]. PTP, considered as an example in this work, is a master slave algorithm. A slave is considered synchronized if the difference between its time and a grand master (GM) time is below the boundaries set, $offset_{max}$ (light-green dotted lines in Fig. 1). An offset is calculated and a correction is performed periodically, i.e., each resynchronization interval (RI). According to PTP, the offset is calculated via timestamps obtained during a message exchange

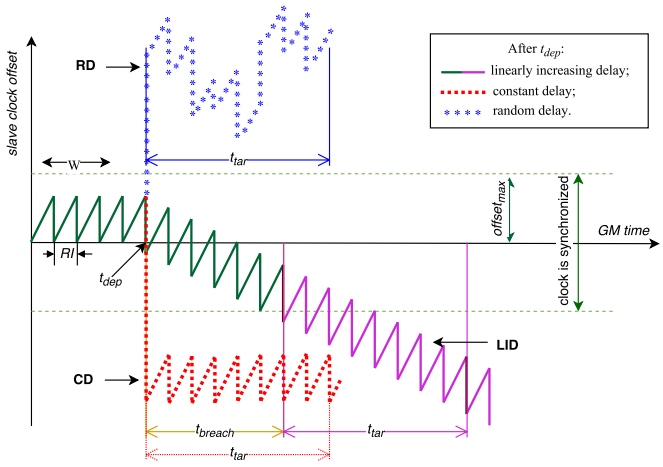

Fig. 1. Clock synchronization under different types of a delay attack

between a slave and a GM. IEEE 1588 can be breached by a delay attack, i.e., if an adversary delays one of the messages used to calculate the offset, it will change the value of the calculated offset, and thereby causing a "correction" into a wrong value. As time is relative for the individual node, it will not realize that the offset is incorrect and thereby it will be brought into unsynchronized state. Clock synchronization protection and the enhancement of current standards is considered by many researchers. For instance, Gaderer et al. [7] developed general guidelines for making the 1588 standard fault-tolerant, e.g., by on-the-fly-time-stamping and allowing asymmetrical delays. A delay attack is identified as a general threat towards time protocols in packet-switched networks [8]. One of the ways to cope with a delay attack is a multipath strategy [9]. However, this approach will require changing the standard and also introduces additional communicational overhead. We have proposed another method that can work on top of the standard and complement it to provide security guarantees for clock synchronization which is essential for safety-critical applications [2]. A monitor approach was then developed to cope with delay attacks.

III. Monitor and Adversary Models

To conduct a sensitivity analysis and determine dependencies between indicator thresholds, we need appropriate models of the monitor and the adversary. In addition, we need a specific use case to define the set of indicators to be monitored.

A. Monitor Model

The monitor considered in this work consists of three network states as demonstrated in Fig. 2. Normal State (NS) is the usual operational state of the system. Quarantine State (QS) is a state used when there are suspicions about safety or security being jeopardized, but not confirmed. Finally, Anomaly Detected State (ADS) is a state used when the monitor is assured that the network is under attack or that there is a failure that could lead to a hazard. The monitor is capable of switching states based on a set of inputs, or so-called indicators. An indicator is a value obtained by monitoring that has two thresholds, a low and a high. Indicators are separated

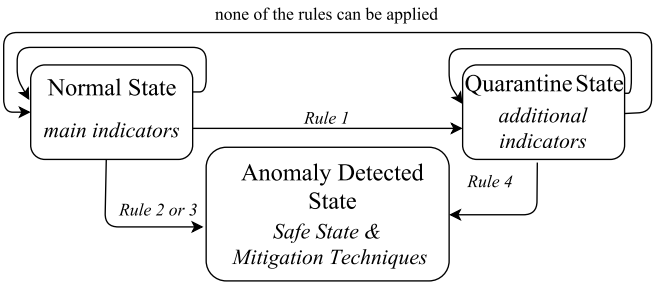

Fig. 2. Network states and connections between them

into two groups: main indicators that are monitored constantly, and additional indicators that are checked only in QS. Thus, the following rules are used to make a decision, Fig. 2:

Rule 1. If a certain number, K_{QS} of the main indicators are above their corresponding low threshold, the system is switched into QS.

Rule 2. If more than K_{QS} of the indicators are above their low thresholds, the system is switched into ADS.

Rule 3. If a certain number K_{ADS}, where $K_{QS} < K_{ADS}$, of their indicators is above the corresponding high threshold, the system is switched into ADS.

Rule 4. If any of the cases described below is true, the system is switched from QS mode to ADS:

1) If there are positive indicators of network anomaly according to the additional indicators deployed in QS;
2) If *Rule 2* or *Rule 3* can be applied.

B. Adversary Model

In this work we consider three types of imposed delay. For each type we formulate the attack space, i.e., the considered values of delays with respective probabilities. The three different types of delay attacks when deployed at t_{dep}, are presented in Fig. 1. During the attack, a delay is imposed every RI, thus different types of delay attacks refer to how the imposed delay varies from one RI to another. A Constant Delay (CD) is the case when the delay is the same, d_{CD} in each RI. When the delay is increasing each RI in a linear fashion, d_{LID}, it is called a Linearly Increasing Delay (LID). Finally, a Random Delay (RD) is a situation when the delay value, d_{RD}, is chosen randomly from an interval of values allowing to bring a targeted node into a unsynchronized state.

• *CD*. We assume that the value of CD is chosen from a predefined set of delays with size N_{CD}:

$$D_{CD} = \{d_{CD,j}\}_{j=1}^{N_{CD}}, \qquad \sum_{j=1}^{N_{CD}} p_{CD,j} = 1. \tag{1}$$

• *LID*. In case of LID we define a set of possible steps, therefore the value of the delay at a particular RI depends on the chosen step and a number of RI, *i*:

$$D_{LID} = \{i \cdot d_{LID,j}\}_{j=1}^{N_{LID}}, \qquad \sum_{j=1}^{N_{LID}} p_{LID,j} = 1. \tag{2}$$

• *RD*. Values for RD are chosen according to a uniform distribution from an interval, where even the lowest value brings a node into unsynchronized state, i.e., it is bigger than $2 \cdot$ *offset*$_{max}$:

$$D_{RD} \in [d_{RD}^{min}, d_{RD}^{max}], \qquad p_{RD,d_{RD}} = \frac{1}{d_{RD}^{max} - d_{RD}^{min}}. \tag{3}$$

A delay attack should be persistent since as soon as a delay is not imposed anymore, the algorithm will calculate a correct offset and the clocks will become synchronized again. The time needed for attack consequences to become apparent depends on the particular node being under attack, and therefore the adversary needs to keep the node in unsynchronized state for t_{tar} to reach its goal, i.e., to disrupt the network and cause significant consequences [2]. Also for LID, there is a certain minimum time needed to actually breach clock synchronization, t_{breach}, as it is showed in Fig. 1.

C. Use Case Parameters and Indicators

To identify dependencies between thresholds and use case, we need parameters that are relevant for both. The following parameters are considered:

• *Generic Boundaries: offset$_{max}$, and RI duration*. These two parameters are dictated by an assumed upper limit of the clock drift and node criticality. The first one determines how often the correction should be done so that the clock has the required precision, which in turn is determined by the criticality of the use case, the particular node load and functionality.

• *Window Size: W*. Calculations of offset values are done based on a set of historic values contained within a window. The window is sliding, i.e, in each RI, a new value is added and the oldest one is discarded.

• *Offset Mean: $\overline{\sigma}_{real}$*. This parameter is defined by the network topology and can be estimated if the offset history is available. Thresholds derived from it, needs to be reevaluated after each new network reconfiguration.

• *Nature Distribution Parameter: λ*. We assume that delays occurring due to natural disturbances in the network follows an exponential distribution [10], with the parameter λ.

Several indicators are suitable for clock synchronization monitoring. A node calculates an offset each RI, therefore, its mean and standard deviation can be monitored in order to identify a new trend in these values caused by an attack. Next, the window used to calculate statistics is an indicator too, as it can help to get additional assurance regarding the presence of a new trend, by scaling down the range of samples used for calculation. Finally, if the trend detection problem is formulated using detection theory, the Neyman-Pearson criterion provides one more indicator to consider [2].

Note that the offset considered in an analysis is the real offset, i.e., the value represented by the distance between a line and the x axis in Fig. 1, while an offset that is calculated as a result of the protocol is different (this offset is demonstrated by the vertical parts of time dependency in Fig. 1, i.e, the jump performed each RI). Therefore, an analysis based only on calculated values is limited to a prediction and reasoning of an adversary behavior [1]. To use an outcome of the analysis for attack detection, a reference about a possible real offset is needed, e.g., via a peer-to-peer check of clocks within the same time domain. However, under the assumption of this check

taking place, the analysis of indicators remains the same and the sensitivity analysis is still valid.

IV. A Sensitivity Analysis

This section provides a brief description of the indicators followed by an analysis demonstrating their dependencies with respect to the adversary model.

A. Mean and Standard Deviation

Statistical characteristics of the true offset, that can be obtained from a measured one if there is a drift reference, are the mean and standard deviation. These can be used as indicators and calculated using a window *W*, as it is presented below:

$$\overline{\sigma} = \frac{1}{W}\sum_{i=1}^{W} offset_i, \tag{4}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{W}(offset_i - \overline{\sigma})^2}{W-1}}. \tag{5}$$

According to the approach described in III-A, there should be two thresholds for every indicator, one lower and one higher. For the mean and standard deviation, the method to set these thresholds is to investigate their attack component, i.e., the component of an indicator value caused by the delay imposed by an adversary, and based on use case characteristics define at which RI the attack needs to be detected after being deployed. To set thresholds, the part of the indicators that are caused by an attack component of the offset can be separated [2]. For simplicity, we assume that the attack component of the offset before the attack is deployed, is equal to zero:

$$\overline{\sigma}_{attack} = \frac{1}{W}\sum_{i=1}^{W} offset_{attack,i}, \tag{6}$$

$$\sigma^2_{attack} = \Big|\frac{1}{W-1}\sum_{i=1}^{W}\Big[(\overline{\sigma}_{attack,i} - offset_{attack,i})^2 + \\ +2\overline{\sigma}_{real,i}\cdot\cdot offset_{real,i}\Big(\frac{\overline{\sigma}_{attack,i}}{offset_{real,i}} - \frac{offset_{attack,i}}{offset_{real,i}} - \\ -\frac{\overline{\sigma}_{attack,i}}{\overline{\sigma}_{real,i}} + \frac{offset_{attack,i}}{\overline{\sigma}_{real,i}}\Big)\Big]\Big|. \tag{7}$$

Eq.(6–7) show that the attack components of both indicators are proportional the value of the delay, as $offset_{attack,i}$ is proportional to the delay imposed at the i-th RI. Therefore, the choice of the range of delays, i.e., its minimum and maximum values, affects the outcome of thresholding. The range of delays should be dictated by the use case specification, e.g., for CD, the range should be starting from a value close to the minimum one required to break clock synchronization, i.e., $4\cdot$ $offset_{max}$, and should be going up till when a further increase does not bring any significant change. For instance, if the network has time-slot based channel access, a node being in a unsynchronized state will start accessing the wrong slot. Once this happens, a further increase of the value of the offset does not matter as such. The attack components of both indicators also depend on the window size, however, this dependency is exploited by considering the window size as an indicator.

The attack component of the standard deviation depends on $\overline{\sigma}_{real}$. Therefore each time a route of the messages is reconfigured, the thresholds have to be recalculated. Moreover, as $\overline{\sigma}_{real}$ depends on λ, the thresholds for the standard deviation should be recalculated in case of using the system in a drastically different environment.

B. Neyman-Pearson Criteria and Detection Coefficient

The problem of attack detection, i.e., a detection of an offset attack component, can be formulated though detection theory and in particular the Neyman-Pearson lemma. The threshold obtained based on the Neyman-Pearson criterion we call Neyman-Pearson threshold [2]. In order to derive a high and a low threshold, the latter needs to be scaled with a trust coefficient, e.g., 60% for the low threshold and 100% for high. The Neyman-Pearson threshold, T_{NP}, can be calculated via a detection coefficient [1] for a particular type of delay attack based on an adversary model, e.g., for LID it can be calculated as follows:

$$T_{NP} = \Big(\sum_{j=1}^{N_{LID}} (e^{\lambda}\cdot i\cdot d_{LID,j}\cdot p_{LID,j})\Big)^W. \tag{8}$$

Eq.(8) is transformed further in a form allowing to determine how the threshold depends on the size of the considered delay space, i.e., N_{LID}. For simplification, we assume that the probability of choosing one of the delays from the set is uniform, i.e, that $p_{LID,j} = \frac{1}{N_{LID}}$, as there are no obvious reasons to prefer a specific one. To assess the difference between situations when the two various sets of delays are considered, we compare the thresholds for the two following sets: (*i*) *set* 1 with delays of size N; (*ii*) *set* 2 with delays of size $N+1$ containing all delays from the previous set and one more value, $d_{LID,N+1}$. A comparison of thresholds for these two cases boils down to the following inequality:

$$(1+\frac{1}{N})\cdot(e^{d_{LID,1}-d_{LID,N+1}} + \ldots + e^{d_{LID,N}-d_{LID,N+1}}) \lessgtr 1. \tag{9}$$

If Eq.(9) is solved with *"less"*, then adding $d_{LID,N+1}$ into consideration lowered the threshold, whereas if it is solved with *"greater"*, then oppositely the threshold was increased. Eq.(9) cannot be unambiguously solved in a general way, however we can point out several cases when it is possible. If $d_{LID,N+1}$ is greater than all delays from the *set* 1, Eq.(9) is solved with *"greater"*. If $d_{LID,N+1}$ is less than all delays from *set* 1, Eq.(9) is solved with *"less"* for a large enough N. In other words, adding a delay that is above the maximum one from the already considered set, will increase the threshold; adding a delay that is below the minimum one from the already considered set, will lower the threshold. The outcome of the intermediate cases depends on N and the particular value of the delay. However, a similar trend to the one described in Sec. IV-A can be observed, namely that when choosing a set of delays to consider, it is important to define a valid range of delays. This threshold dependency from the natural distribution is quite obvious, as the greater λ is, the higher the threshold is, which correlates with the fact that the more network disturbance, the higher the malicious deviation needs to be in order to be detected.

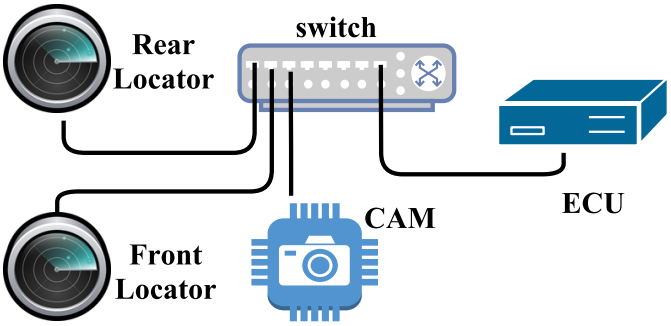

Fig. 3. A Collision Avoidance System

The case of CD can be analyzed in a similar fashion, as its detection coefficient is similar. Therefore, we only consider the RD case below. The threshold for the RD case can be calculated based on a detection coefficient for RD:

$$T_{NP} = \left(\frac{e^{d_{RD}^{min}}}{(d_{RD}^{max} - d_{RD}^{min})} \left(e^{\lambda \cdot (d_{RD}^{max} - d_{RD}^{min})} - 1 \right) \right)^{W}. \quad (10)$$

If the range of RD is fixed, i.e., $[d_{RD}^{max} - d_{RD}^{min}] = const$, then the greater the minimum value, the higher the threshold. If the minimum delay is fixed, i.e., $d_{RD}^{min} = const$, then the threshold's dependency from the range of delays has the same character as $f(x) = \frac{e^x - 1}{x}$, where $x > 0$. For the defined range of x values, this function is constantly growing and has a limit equal to 1 at $x = 0$. Therefore, for this case we have a minimum threshold that equals $e^{d_{RD}^{min} \cdot W \cdot \lambda}$. The greater the range of delays, the higher the threshold, and the same goes for the dependency of λ.

C. Window Size

Having the window size as an indicator was initially proposed as an additional indicator deployed only in QS. This indicator is used in addition to the ones considered above, as when varying the window size, the monitor still checks the mean, standard deviation or the Neyman-Pearson criterion. Therefore, all dependencies identified for the other indicators are relevant for the window size as well.

V. A Use Case Specific Sensitivity Analysis

An example of an CPS is considered to illustrate thresholds dependencies for a particular use case.

A. Use case — a Collision Avoidance System

A Collision Avoidance System based on a time-triggered architecture is an example of a CPS from the automotive domain and used to investigate the dependencies of the indicators with respect to a particular use case. As it is shown in Fig. 3, the system consists of (*i*) a switch managing connections and following a schedule, (*ii*) a camera, CAM, located in the front part of a vehicle, (*iii*) two locators - one in the front, and one in the rear part of the vehicle, and (*iv*) an Electronic Control Unit (ECU). The camera sends pictures of the road in front of the vehicle to the ECU, so that the latter can estimate a situation. The ECU also gets information from locators, that estimate the distance from the vehicle to the nearest object in a certain direction. Locators send data only if an object is detected within a specified range. The camera in its turn sends data constantly within specified time-slots. Once the ECU makes a decision that the current situation is potentially dangerous, it sends a signal to the outer dependency of this system, e.g., a steering control system. The system has a time-triggered architecture, i.e., channel access is time-based and there is a schedule for data transactions. Nodes in the system are using the IEEE 1588 standard for clock synchronization. We assume that a time slot duration is 100 μs [11], RI is 50 *ms* and that a clock drift is 10^{-2} s/s, i.e., up to 0.1 μs deviation every 1 *ms*. Therefore, the synchronization boundaries, $offset_{max}$, can be set with a 2 μs safety gap and be equal to 7 μs.

B. Use Case Dependencies

The particular use case along with the choice of communication structure, set the boundaries and the RI duration. These values define which delay values can be used to breach clock synchronization, e.g., the minimum delay for CD and RD or a breaching time for the LID attack. In the considered example, the minimum delay value for CD is greater than $4 \cdot offset_{max}$, i.e., 28 μs. The maximum delay that makes sense to consider is defined by the time-slot duration. From an adversary perspective, once the offset exceeds the time-slot duration, there is no difference to further exceed. The node functionality sets the time that the node should be in a unsynchronized state for an adversary to succeed, i.e., t_{tar}. In the considered example, the camera sends information with a high frequency, as a situation on the road can change very fast considering the vehicle speed. Therefore, to have significant consequences, it is enough to miss one time-slot during a vehicle operational phase. We can assume that the camera messages are scheduled every other time-slot, thus being in a unsynchronized state for more than 100 μs is not acceptable for the system to fulfill its functional requirements. Locators are using their time-slots (every other that is not occupied by the camera), only if there is an object detected. Consequently, if there is no detected object, i.e., in case of driving during late night on a highway, a locator in a unsynchronized state would not be noticed by the system. Therefore, for locators t_{tar} depends on the operational situation, but this on average is longer than for the camera.

VI. Discussion

The conducted analysis regarding an adversary model, shows that it is a necessity to determine valid limits for the values of the imposed delays. This fact implies a necessity to know a particular use case in order to specify an adversary model. A threat model includes an adversary model as well as a use case specification [12]. However, to calculate thresholds for when the monitor should switch states, requires a deeper level of use case specification, i.e., the network topology should be specified together with node functionalities. In the best case, a schedule is provided such that the node communication pattern and frequency of sending and receiving messages can be estimated. Moreover, as it was demonstrated in Sec. IV, the standard deviation attack component depends on the offset mean, which in turn depends on a route between a GM and a slave used for PTP protocol message exchange. This

limits the flexibility of the method, although it is reasonable to expect a security solution being tightly use case dependent. A way to improve the flexibility is to automate the calculation of thresholds or cardinally relax this connection by imposing requirements on the degree of indicator inter-dependency in the process of indicator development.

Another indirect outcome of the analysis is that it provides grounds for an indicator generalization. By analyzing a specific use case, we can identify which indicators are relevant and which can be used efficiently as main or as additional indicators. As identified in Sec. IV, indicator thresholding is heavily connected to the use case and the adversary model. However, the indicator applicability relates not to a particular value of a threshold, but to a trend in indicator behavior, which is identified by its general analysis [2]. Therefore, we can conclude that the results on indicator applicability can be generalized, but not their effectiveness, which should be investigated for each particular use case. For instance, as standard deviation thresholds require knowledge about the communication link between a GM and a slave, it can be used as an additional indicator, since its threshold calculation can be done only when entering the QS and thus a network reconfiguration during the NS would not require its recalculation. The mean in turn requires less amount of information about the specific use case and adversary model, and therefore it is a good candidate for being a main indicator. Finally, the Neyman-Pearson criterion is in between mean and standard deviation regarding required input, as it does depend on the adversary model, but does not require data about the particular communication link. Thus, this indicator can be used in as both states.

The monitor analyzed in this work is a type of Intrusion Detection System (IDS) [13], which can be used for a general application [14]. Even though the proposed monitor use case is narrowed down to clock synchronization handling here, it is possible to extend it to cover other assets of CPSs following a similar approach, e.g., to develop new relevant indicators or to re-evaluate the applicability of already developed ones. The considered indicators and the different ways to threshold them are based on simple equations, and therefore we analyzed them using standard mathematical approaches and do not formalize the analysis further [15]. Risk evaluation and cost assessment of the monitor [16] are outside the scope of this work, however it can be a logical step in further investigations of its development. The basics of the monitor strategy from a risk and cost perspective were considered in [2], thus possible work extensions can be built upon the identified requirements.

VII. Conclusions

The analysis conducted in this work focuses on indicators and thresholds used by a monitor for switching states when it detects that clock synchronization is under attack. The analysis shows a strong dependency on some particular values of thresholds derived by the use case and the adversary model. For the Standard Deviation indicator, thresholds for its attack component also depend on the network topology, implying that this indicator is best used as an additional one, as then its threshold recalculation is not needed during NS in case of network reconfiguration. Thresholding the Mean indicator requires less details about the network, and therefore it can be used as a main indicator. Neyman-Pearson thresholding has a higher level of dependency on the use case compared to Mean but less compared to Standard Deviation, therefore it can be used in both roles and assigned depending on other factors that are outside the scope of this work, e.g., a risk assessment.

Future work includes detection rate calculation to complete the evaluation of the monitor, investigation of how the proposed set of indicators can be used and their effectiveness for other clock synchronization protocols. Also an investigation on more effective ways of detecting offset perturbations with a significantly small slope should be conducted.

Acknowledgments

The research leading to these results has received funding from The Knowledge Foundation through the SIDUS project 20130086 READY and from the SafeCOP project funded from the ECSEL Joint Undertaking under grant agreement n692529, and from National funding.

References

[1] E. Lisova, E. Uhlemann, W. Steiner, J. Åkerberg, and M. Björkman, "Game theory applied to secure clock synchronization with ieee 1588," in *Proceedings of ISPCS*, Sept 2016, pp. 1–6.

[2] E. Lisova, E. Uhlemann, J. Åkerberg, and M. Björkman, "Delay attack versus clock synchronization - a time chase," in *Proceedings of ICIT*, March 2017, pp. 1136–1141.

[3] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, pp. 161–166, 2011.

[4] "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.

[5] A. Treytl and B. Hirschler, "Security flaws and workarounds for ieee 1588 (transparent) clocks," in *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, Oct 2009, pp. 1–6.

[6] M. Ullmann and M. Vögeler, "Delay attacks - implication on ntp and ptp time synchronization," in *Proceedings of ISPCS*, Oct 2009, pp. 1–6.

[7] G. Gaderer, H. Muhr, M. Horauer, and T. Sauter, "Extending ieee 1588 to fault tolerant clock synchronization 2004," in *Proceedings of the WFCS*, 2004, pp. 353–359.

[8] T. Mizrahi, "Security Requirements of Time Protocols in Packet Switched Networks," RFC 7384, Oct. 2014.

[9] ——, "A game theoretic analysis of delay attacks against time synchronization protocols," in *Proceedings of ISPCS*, Sept 2012, pp. 1–6.

[10] S. H. Lin, T. C. Lee, and M. F. Gardina, "Diversity protections for digital radio-summary of ten-year experiments and studies," *IEEE Communications Magazine*, vol. 26, no. 2, pp. 51–63, Feb 1988.

[11] W. Steiner, "An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks," in *2010 31st IEEE Real-Time Systems Symposium*, Nov 2010, pp. 375–384.

[12] E. Lisova, E. Uhlemann, J. Åkerberg, and M. Björkman, "Towards secure wireless ttethernet for industrial process automation applications," in *Proceedings of ETFA*, Sept 2014, pp. 1–4.

[13] M. Garuba, C. Liu, and D. Fraites, "Intrusion techniques: Comparative study of network intrusion detection systems," in *Proceedings of ITNG 2008*, April 2008, pp. 592–598.

[14] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks," *Expert Systems with Applications*, vol. 29, no. 4, pp. 713 – 722, 2005.

[15] H. H. Feng, J. T. Giffin, Y. Huang, S. Jha, W. Lee, and B. P. Miller, "Formalizing sensitivity in static analysis for intrusion detection," in *Proceedings IEEE SSP*, May 2004, pp. 194–208.

[16] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *J. Comput. Secur.*, vol. 10, no. 1-2, pp. 5–22, Jul. 2002.