# Information Hiding in Cyber Physical Systems: Challenges for Embedding, Retrieval and Detection using Sensor Data of the SWAT Dataset

Kevin Lamshöft*
Christian Krätzer*
Jana Dittmann*
firstname.lastname@iti.cs.uni-magdeburg.de
Research Group Multimedia and Security
Otto-von-Guericke University
Magdeburg, Germany

Tom Neubert†
Claus Vielhauer†
firstname.lastname@th-brandenburg.de
Brandenburg University of Applied Sciences
Brandenburg an der Havel, Germany

## ABSTRACT

In this paper, we present an Information Hiding approach that would be suitable for exfiltrating sensible information of Industrial Control Systems (ICS) by leveraging the long-term storage of process data in historian databases. We show how hidden messages can be embedded in sensor measurements as well as retrieved asynchronously by accessing the historian. We evaluate this approach at the example of water-flow and water-level sensors of the Secure Water Treatment (SWAT) dataset from iTrust. To generalize from specific cover channels (sensors and their transmitted data), we reflect upon general challenges that arise in such Information Hiding scenarios creating network covert channels and discuss aspects of cover channel selection and and sender receiver synchronisation as well as temporal aspects such as the potential persistence of hidden messages in Cyber Physical Systems (CPS). For an empirical evaluation we design and implement a covert channel that makes use of different embedding strategies to perform an adaptive approach in regards to the noise in sensor measurements, resulting in dynamic capacity and bandwidth selection to reduce detection probability. The results of this evaluation show that, using such methods, the exfiltration of sensible information in long-term scaled attacks would indeed be possible. Additionally, we present two detection approaches for the introduced hidden channel and carry out an extensive evaluation of our detectors with multiple test data sets and different parameters. We determine a detection accuracy of up to 87.8% on test data at a false positive rate (FPR) of 0%.

*Kevin Lamshöft provided the idea, concept and realisation of the presented covert channel and contributed to the paper in a whole. Christian Krätzer contributed to Section 1 and 2 and gave the initial idea for the proposed synchronisation method. Jana Dittmann contributed to the overall structure by improvements in the systematical structuring of the findings and contributions, in the formalization of the approach, the discussion of embedding and retrieval keys and Figure 1.
†Tom Neubert contributed the idea, concept and evaluation of detection approaches in Sections 5 and 6. Claus Vielhauer contributed to the overall structure of the paper, Figure 1 and the general design of detection and evaluation in Sections 5 and 6.

*IH&MMSec '21, June 22–25, 2021, Virtual Event, Belgium.*
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8295-3/21/06.
https://doi.org/10.1145/3437880.3460413

## CCS CONCEPTS

• **Security and privacy** → **Network security**; **Systems security**; **Intrusion detection systems**; **Security in hardware**; **Firewalls**; **Systems security**; **Intrusion detection systems**; • **Applied computing** → **Industry and manufacturing**; • **Social and professional topics** → **Computer crime**; **Automation**; **Computer crime**; • **Theory of computation** → **Cryptographic protocols**.

## KEYWORDS

Information Hiding; Steganography; Covert Channels; Network Covert Channels; Cyber Physical Systems; Industrial Control Systems; Sensors; Steganalysis; Detection; Warden

**ACM Reference Format:**
Kevin Lamshöft, Christian Krätzer, Jana Dittmann, Tom Neubert, and Claus Vielhauer. 2021. Information Hiding in Cyber Physical Systems: Challenges for Embedding, Retrieval and Detection using Sensor Data of the SWAT Dataset. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security (IHMMSec '21), June 22–25, 2021, Virtual Event, Belgium.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3437880.3460413

## 1 INTRODUCTION

Information Hiding in the field of network steganography has recently attract more attention, as targeted attacks have begun to use stealth mechanisms to avoid early detection, especially in the context of so called Supply Chain Attacks like the recent SolarWinds attacks [1]. As recent publications show, Information Hiding acts as a novel threat for Industrial Control Systems (ICS) and Cyber Physical Systems (CPS) in general, as they enable attackers to infiltrate systems, exfiltrate valuable information and establish command and control channels in such a stealthy way that common intrusion detection and prevention systems are likely to fail (i.e. do not reliably detect such attacks [14]). Another application of (Network-) Information Hiding is to avoid detection of lateral movement in an attacked domain after the first initial intrusion, and can especially be used to circumvent firewalls and other means of network segregation (incl. air gapping). To establish stealthy covert channels, one common way in Network Information Hiding is to identify fields in network protocols which usually contain high entropy content and then modify these to embed hidden messages [10].
In this paper, we evaluate the feasibility of four selected embedding methods to hide information in the transmission of sensor readings

at the example of water-flow and water-level sensors in an ICS environment. We leverage the circumstance that such systems are commonly designed similar to the Purdue Reference Architecture [26], which employs the transmission of process data to a centralized database (often referenced to as process- or data historian) for long-term storage. Additionally, we evaluate how such communication flows can be used to establish protocol-agnostic covert channels for information exfiltration and generate data sets for evaluations based on the introduced embedding methods. Furthermore, we introduce novel detection approaches to detect such attacks. These detection approaches are extensively evaluated in regards to their performance on data sets generated with different embedding methods and bandwidth.
The contributions of our paper can be summarized as follows:

- Proposal for establishing covert channels for data exfiltration using sensor data collected in ICS historians (long-term storage of process data)
- Design of a protocol-agnostic covert channel that is asynchronous and long-term persistent
- Discussion on temporal persistence of hidden messages
- Discussion on selection of appropriate embedding strategies
- A practical evaluation of hiding information in the noise of sensor values of water-flow and water-level sensors in the Secure Water Treatment (SWAT) dataset [6]
- Generation of evaluation data for testing & detection approaches
- Design and implementation of novel detection approaches for the introduced covert channels
- An extensive evaluation of the introduced detection approaches to determine their performance
- Analysis of detection approaches for integration in ICS

The work is structured as follows: In Section 2 we discuss the state of the art regarding network steganography and corresponding covert channels related to our work. In Section 3 we describe the threat model, attack scenario and attacker model which represents the basis for our exfiltration strategy, which we discuss in detail in Section 4. Section 5 introduces our novel detection approaches based on [10] and how it could be used in a real world ICS. In Section 6 the evaluation of our detection approaches is summarised, including evaluation goals, data sets and results. Section 7 concludes our work and presents a perspective on potential future work.

## 2 STATE OF THE ART

Compared to research on steganographic methods for media objects (esp. digital images), network steganography has received much less attention in the past decades. It has always been around since the digital domain started to gain pace, with first noticeable publications in the 1980s (see e.g. [18]), 1990s ([8]), early 2000s ([17]), but those have been small in numbers and mostly focusing on more or less trivial storage channels (like e.g. the embedding into unused TCP/IP header fields discussed in [17]). With the advent of covert timing channels in network steganography around 2005 (see e.g. [22] or [21]) the field gathered some noticeable increase in research interest. If it comes to the task of performing steganalysis on a hidden channel created by network steganography, then this task relates to the paradigm of performing pooled steganalysis on batch steganography. This concept, introduced in 2007 in [11] focuses on the problems of hiding information securely when spreading it across a batch of covers as well as corresponding considerations on the detection of such embedding. In his seminal work, Andrew D. Ker illustrated the paradigm with a case study using LSB replacement in large batches of images. While large bursts of media objects might create some suspicion, such an occurrence in network traffic is in many cases considered normal. Therefore, it is an obvious choice to investigate into such methods for network steganography. Based on this realisation and driven by increasing numbers of Malware relying on steganographic methods (for purposes of hidden infiltration, command and control communication or data exfiltration) since 2011, the field of research in network steganography and steganalysis has received a huge boost as well as a shift in focus. While early approaches have been looking into generic network protocols such as TCP/IP or HTTP for cover channels, more recent publications have started looking into protocols that would be much more relevant in the industrial espionage or sabotage scenarios targeted by modern Malware families. As a result network protocols used for automation networks, found in Industrial Control Systems or automation systems in general are of special interest. In [25] Wendzel et al. use an analogy from nature to describe Information Hiding in Cyber Physical Systems. The authors of that paper use hoarding and caching tactics found in the behaviour of animals to describe how attackers potentially could use Information Hiding to establish secret *steganographic data storage* in smart buildings. Their focus lies on actuators and unused registers of automation systems in smart buildings which are used as secret storage to hide information. This idea of distributing a hidden message in multiple covers is referred to as *scattered hoarding*. In their work they focus on *steganographic data storage* in Cyber Physical Systems by using a smart buildings automation system as a *secret storage*. This is framed by a scenario in which two spies want to secretly exchange information in an asynchronous way by using a *dead drop*, which in this case is the smart building. The main issue with that approach is that the hidden data might be overwritten by control processes or user interaction. Since ICS usually deploy long-term storage of historic process data, we propose to leverage this fact by encoding hidden messages by manipulating sensor or actuator values and retrieve those by accessing the long-term storage. In [9] and [4] Herzberg and Ahmed manipulate the noise levels in senor data to establish a covert timing channel between sensors and actuators. Other work evaluates covert channels found in common automation protocols, e.g. Modbus/TCP [13] [15], OPC/UA [10] [5] and MQTT [24]. Additional work modulating sensor values to perform attacks on Industrial Control Systems include the works of Krotofil [12] and Ahmed [4] [6] [3]. However, certain publications regarding covert channels in common IT networks might be applicable as well in industrial networks.

## 3 THREAT MODEL

As described in the previous section, in contrast to [25] we use an attack scenario, in which we leverage the long-term storage of process data as as a cover channel for exfiltrating data out of

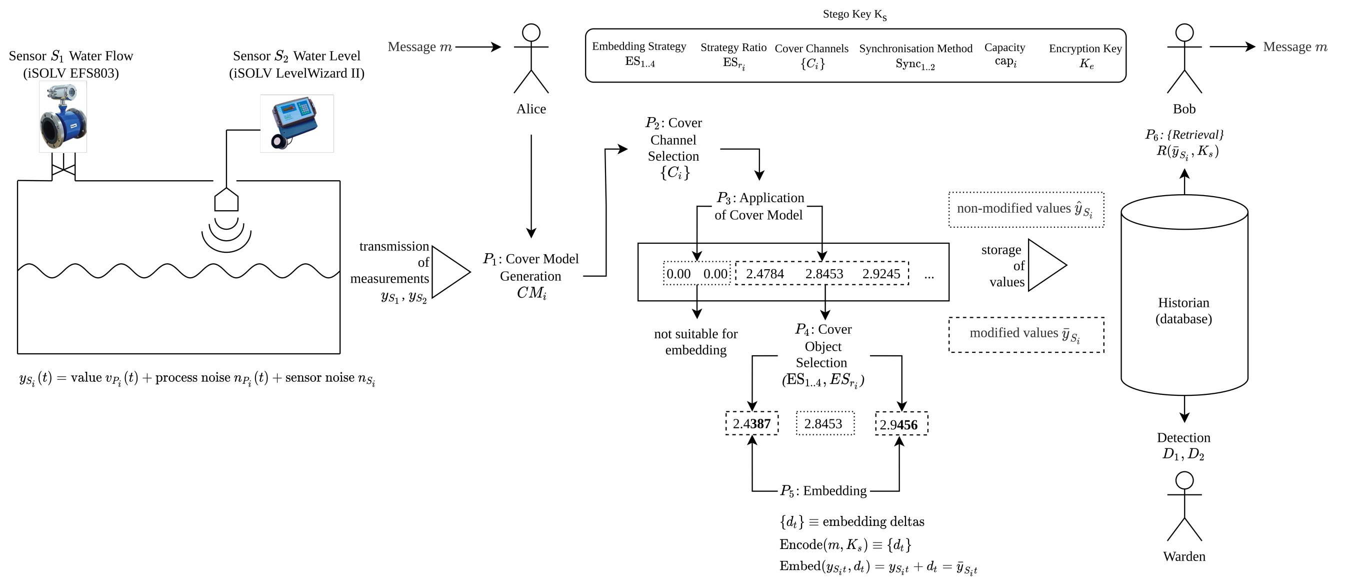
Sensor $S_1$ Water Flow (iSOLV EFS803)
Sensor $S_2$ Water Level (iSOLV LevelWizard II)
$y_{S_i}(t)$ = value $v_{P_i}(t)$ + process noise $n_{P_i}(t)$ + sensor noise $n_{S_i}$
transmission of measurements $y_{S_1}, y_{S_2}$
Message $m$
Alice
$P_1$: Cover Model Generation $CM_i$
Stego Key $K_S$
Embedding Strategy $ES_{1..4}$
Strategy Ratio $ES_{r_i}$
Cover Channels $\{C_i\}$
Synchronisation Method $Sync_{1..2}$
Capacity $cap_i$
Encryption Key $K_e$
$P_2$: Cover Channel Selection $\{C_i\}$
$P_3$: Application of Cover Model
0.00 0.00 2.4784 2.8453 2.9245 ...
not suitable for embedding
$P_4$: Cover Object Selection $(ES_{1..4}, ES_{r_i})$
2.4387 2.8453 2.9456
$P_5$: Embedding
$\{d_t\} \equiv$ embedding deltas
Encode$(m, K_s) \equiv \{d_t\}$
Embed$(y_{S_it}, d_t) = y_{S_it} + d_t = \bar{y}_{S_it}$
non-modified values $\hat{y}_{S_i}$
storage of values
modified values $\bar{y}_{S_i}$
Historian (database)
$P_6$: {Retrieval} $R(\bar{y}_{S_i}, K_s)$
Bob
Message $m$
Detection $D_1, D_2$
Warden

**Figure 1: Our approach for embedding and retrieval in sensor measurements.**

an Industrial Control Systems (ICS) network. In the following sections, we describe our attack scenario as well as the corresponding attacker model.

## 3.1 Attack Scenario

In order to describe our attack scenario we use the MITRE Att&CKics framework [16], which is a comprehensive knowledge base and allows for modelling realistic attack scenarios and attack procedure descriptions for ICS. In our attack scenario we assume that the attacker wants to exfiltrate sensible information from the production network, like firmware versions, IP-addresses or details on the process itself or its logic. While the focus is set on the exfiltration part here for simplicity, the proposed covert channel is geared to be used as part of a more complex Command & Control (C&C) channel as well (when paired with a covert infiltration channel). To be able to exfiltrate such information, we assume that on the embedder side a device is compromised which takes part in the communication flow between sensors gathering data about a physical process which is collected and processed by Programmable Logic Controllers (PLCs) and are (with potential inter-stops) aggregated and stored in a process database, or historian. Depending on the network architecture, those components could be PLCs, network elements (e.g. firewalls), Human-Machine-Interfaces (HMIs) or other components, like data aggregators or protocol converter. On the receiver side, each device which in some way has (potential) access to the historical data, like the database server itself or engineering workstations, are potential receiving/retrieving devices, which need to be compromised in a way that the hidden message can be extracted from the historian. For example, an engineer who has access to the historical data can act as inside threat, by e.g. infecting her computer with a malware which is capable of retrieving the hidden information or writing such code directly at her workplace. In the Att&ckics framework this scenario is part of the *Initial Access* phase and is reflected in the tactics of *Data Historian-*, *Engineering Workstation-* and *Supply Chain- Compromise*. The previously described scenario of the inside threat, developing the malware at the workplace is currently not reflected in the Att&icks framework.

## 3.2 Attacker Model

For our scenario the knowledge level of the attacker only plays a minor role, as the aim of the data exfiltration strategy is to gain more insight and might be used as part of the reconnaissance for further attacks. Nevertheless, in order to establish the covert channel, prior infections are needed, which require a certain set of a priori information. For our attacker model, we assume that the attacker has the capabilities to deploy the tactics as described before, namely supply chain attacks and engaging inside threats. As those tactics require high efforts, they are deployed by actors with sufficient resources, like nation state actors or advanced persistent threats (APT). Therefore, our attack scenario and attacker model is based on the assumption of a highly targeted attack with nearly unlimited resources regarding time, money and personnel. On the technical side though, we encounter limited resources and capabilities as limitations occur regarding computational power and data storage, when the embedding and retrieval takes place in a stealthy way on components with limited resources, like PLCs or firewalls.

## 3.3 Experimental Setup

For the experimental evaluation we use the Secure Water Treatment (SWAT) dataset from iTrust [6] as it provides a large amount of real process data, which allows for a realistic simulation of our attack scenario. We are using a subset from the A1 and A2 collection from December 2015 and focus on two sensors, water flow (`FIT101` in the original dataset, here $S_1$) and water level (`LIT101` in the original dataset, here $S_2$). These sensors are selected as they represent two common but distinctive types found in such systems. For this evaluation, the process data directly from the sensors is used in order to simulate a scenario, in which PLC would modulate the received values. Figure 1 illustrates this attack scenario. The focus here is to evaluate the performance of four different embedding strategies (see Section 4) for each sensor ($S_1, S_2$) and the resulting effect on the detection. For the data generation, a subset of 24855 datapoints is selected from the SWAT dataset, which reflects around 6 hours of the physical process. The first two hours (8285 values) are used to build the cover model (see Section 4.1) for each sensor (and can be used as a non-stego set), the other 4 hours (16570 values) are used for embedding.

## 4 EMBEDDING & RETRIEVAL

As shown in Figure 1, our approach for embedding and retrieval is based on six Process Steps $P_i$, which can be summarized as follows: $P_1$: Cover Model Generation, $P_2$: Cover Channel Selection, $P_3$: Cover Model Application, $P_4$: Cover Object Selection, $P_5$: Embedding, $P_6$: Retrieval. In the following we use these as a guideline and describe each phase in more detail.

### 4.1 Cover Model Generation ($P_1$)

Sensor measurements can have very different characteristics depending on the type of sensor, the physical process that is monitored and the current state of that process. For example, in the case of the used water flow sensor $S_1$, a constant 0.00 is returned when no water is passing the sensor. Obviously, it is not possible to plausibly embed in such values. Therefore, to be able to decide whether a measurement is suitable for plausible embedding or not, a (cover) model for each sensor is developed by observing measurements over a given period of time, e.g. over the course of day. The resulting cover model $CM_i$ defines constraints which indicate whether the current measurement is suitable for plausible embedding. To derive such constraints, we combine two methods: (1) we use a simple states model to describe the current state of the physical process and (2) we estimate the noise of the sensor for this state. Another important factor is that sensors measurements can show recognizable patterns over time (e.g. reaching a peak level, after filling a water tank). In our approach, we explicitly seek such recognizable patterns to use them for *synchronisation* between covert sender and receiver. These recognizable patterns are used to enhance the cover model and to define constraints, when values are used for synchronisation or embedding the actual message. Figure 2 illustrates the cover models at the example of the aforementioned sensors $S_1, S_2$. As described in the threat model, the attacker might have unlimited resources from an organisational standpoint, but when targeting automation devices like PLCs or HMIs, the attacker only has limited computational resources on that specific target system. Nevertheless, to be able to develop a representative cover model, we leverage the fact, that in general physical processes can be simplified in a states model with two states: steady states and transient states. In steady states values are either constant or fluctuate around a specific value, whereas transient states describe the transition between steady states. By identifying the current state, a simplified model of the physical process is developed by noting the average duration of each phase, its mathematical description

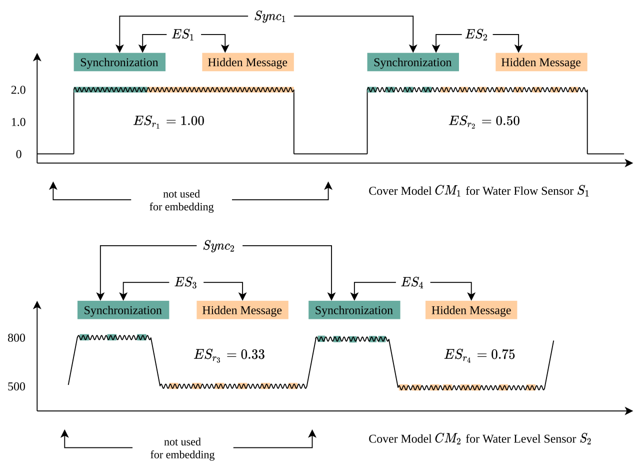

**Figure 2: Cover Model $CM_1, CM_2$, Synchronisation Method $Sync_1, Sync_2$ and Embedding Strategies $ES_1, ES_2, ES_3, ES_4$ for Sensors $S_1, S_2$) (Water Flow, Water Level).**

(static/linear/exponential/logarithmic increase/decrease) and the estimated noise for that state. As shown in Figure 2 for both Sensors $S_1$ and $S_2$ two methods $Sync_1, Sync_2$ are used for synchronisation. In case of the water level sensor $S_2$, we are using the steady state above a water level of 812 centimeters for synchronisation and the linear transient state (which look to be steady, but are linear increasing) around the value of 500 centimeters for embedding the actual message ($Sync_2$). The synchronisation method $Sync_1$ for the water flow sensor $S_1$ is quite different in comparison, as there are two steady states, in which either water is currently flowing or not. If no water is flowing through the sensor, the value is zero and therefore not suitable for embedding. Therefore, embedding is only possible, when water is flowing through the sensor. For synchronisation, we use the transition between not flowing and flowing. The first 128 values after the sensor started to read a flow are used for syncing and the rest for embedding the actual message. In order to get comparable results, we are using the same hidden message for both sensors, as well as the same parameters for the embedding strategies. In addition to that, the states model is helpful as well when employing a a dynamic capacity strategy by calculating the noise per state to use different capacities (hidden bits per cover object) in correlation to the noise, which ultimately might make detection more difficult. In the following the estimation of sensor and process noise is described. As shown in [12] sensors of Cyber Physical Systems can show different types of noise, depending on the measured physical property and type of sensor. The physical process and the sensor itself induce two types of noise [4]: (1) **sensor noise** which is inherited by the (in-)accuracy of a sensor measuring a physical property and (2) **process noise** which are fluctuations of the physical process, e.g. a moving liquid. A sensor measurement $y_{S_i t}$ can be described as the combination of the physical value $v_t$ and the noise $n_t$, which is the sum of the process noise $n_{pt}$ and the sensor noise $n_{S_i t}$ at a given point in time $t$:

$$y_t = v_t + n_t = v_t + n_{pt} + n_{S_i t}$$

To embed a hidden message into the measurement of the sensor, the attacker introduces a $\delta_t$ (which might be positive or negative) to the measured value of the sensor (see Figure 1). The modified value which is transferred and stored in the historian can therefore be described as:

$$\bar{y}_t = y_t + \delta_t = v_t + n_t + \delta_t$$

In order to stay undetected the aim is to keep modifications so small, that a) values are still within the limits and distribution of usual noise and b) still a receiver is able to retrieve the hidden information. In a dynamic capacity scenario the information hider has to solve the optimization problem of finding the highest possible capacity at a given point in time while staying within the limits and distribution of the usual noise. Assuming the usual distribution of the combined noise of sensor and process noise is $n_t \in P$ the distribution of the sensor reading with the embedded message should be within this usual noise distribution $n_t + \delta_t \in P$ to avoid detection. To calculate the noise, we follow a modified approach inspired by [2] and calculate local noise by using the standard deviation on fixed window sizes. In addition to that, we use the distribution of the last digits as a second feature for noise calculation. The window size is selected individually for each sensor, as transient and steady states of the underlying control process might vary in length. To get the potential steganographic capacity per sensor value in bit, we use the doubled standard deviation of the values for a given window $X$ to the logarithmic base of 2 to receive the deviation in bits: $cap_{max}(X) = log_2(2\sigma(X))$. This gives us the maximum amount of bits which can potentially be modified while staying within the usual fluctuations of the values.

## 4.2 Cover Channel Selection ($P_2$)

The amount of potential cover channels is given by the amount of I/Os the covert sender has access to. For example, if the covert sender is a PLC, the list of potential cover channels is given by the sensors and actuators that are connected to the PLC. In case of a compromised network element, like a protocol converter or firewall, the available cover channels are given by the amount of transferred I/O values in the network flow. Therefore, in this paper the number of potential cover channels is two, namely $S_1$ and $S_2$.

As described in Section 2, in [25] the authors propose to use a scatter hoarding strategy to hide a message across multiple unused registers of sensors in a smart building. This concept is driven by the idea to achieve a potentially high bandwidth while staying stealthy as the message is distributed on a large set of cover objects. This discussion of whether to hide a given message in a small number of cover objects (resembles larder hoarding) or to scatter this message across a larger set of cover objects (scatter hoarding) can be found in the concept of batch steganography and steganalysis by Ker in 2007 [11]. In this work it is shown to be the best strategy to embed only in a small number of cover objects - at least if the warden is using a detection method explicitly designed for such pooled steganalysis. This on the other hand leads to the conclusion, that in order to choose a proper strategy for an information hider, he has to keep the warden in mind, which lefts him in a game theoretic problem, as described by Schöttle and Böhme in 2013 [20]. Therefore, the potentially best strategy for the hider is specific to his knowledge about the system and the warden, his capabilities, his position in the system and therefore access to cover objects and cover channels.

In nature, rodents like squirrels recache their food (move it to another cache) in order to prevent pilferage. In the context of Information Hiding, pilferage can be seen as detection and suppression of a covert channel by an active warden. Taking this concept to Information Hiding, two interesting aspects arise: The first aspect we can take from nature, is that hidden elements (in this case a hidden message) may have different lifetimes. A hidden message might be - depending on the on the covert channel - only receivable in real-time, whereas when using a different covert channel a hidden message might be retrievable even years later. This is what we describe as the **temporal aspect** in regards to **persistence** of a covert channel or hidden message. The second aspect we can take from nature is the general concept of recaching which is resembled in Information Hiding as changing the *embedding position* or *cover selection*. Here we can define four major types of **recaching**, with multiple options for each type:

(1) **Cover Channel Selection** (change *cover* channel)
(2) **Covert Channel Hopping** (change *covert* channel)
(3) **Embedding Position Change** (change embedding position within a cover object)
(4) **Cover Object Selection** (select different set of cover objects)

For each type multiple options of operation are possible:

(1) **recache-by-default**: constant change (covert protocol)
(2) **triggered-recache**: triggered (e.g. by control message)
(3) **conditional-recache**: triggered when a pre-defined condition is met e.g. bandwidth below threshold

One of the main concepts in this exfiltration strategy is the fact, that in CPS it is common to store process data for longer periods of time in an historian database. This circumstance is leveraged to establish an asynchronous covert channel, where the hidden information is retrieved independently from the point of time when it got embedded. Therefore, such **asynchronous covert channels** can only be established, if the *cover objects* are *persistent*. Such persistence can be further differentiated. For Cyber Physical Systems we use here three **levels of persistence** (which vary depending on the actual system):

(1) **long-term persistence**, similar to dead drops, e.g. historians (months to years)
(2) **mid-term persistence**, e.g. log files (days to weeks)
(3) **short-term persistence** (minutes to hours)

In contrast to that a **synchronous covert channel** makes use of an **ephemeral cover channel**, e.g. direct TCP/IP communication between an HMI and PLC (under the premise, that communication does not get recorded). In case of the proposed data exfiltration strategy, we are using the long-term persistent storage of process data as a cover channel. But as this differentiation shows, there are potentially a large amount of other similar attack scenarios leveraging different *levels of persistence*. Another application of this differentiation is in the design phase of a warden, as it is important to place wardens in the right positions to be able to detect such covert channels.

## 4.3 Cover Model Application ($P_3$)

The main objective here is to decide for each Cover Channel $C_i$ whether the current cover object (= measurement value $y_{S_i t}$) which is transferred on that channel is used to embed a hidden message, used to synchronise or not used at all (e.g. because the noise is too low). This decision is driven by the selected synchronisation method $Sync_i$ and the cover model $CM_i$ (both described in 4.1), which defines constraints, whether a cover object is suitable to embed a message, and whether it is part of a synchronisation or embedding.

## 4.4 Cover Object Selection ($P_4$)

In nature, animals are searching and using different sized caches and amounts of food per cache to prevent or minimize losses in case of pilferage. While the terminology might vary across different publications, in (Network-) Information Hiding usually two types of (steganographic) payload variation are common:

- **Capacity Modulation**, which is steganographic payload per cover object.
- **Bandwidth Modulation**, which is steganographic payload over time.

These two options are especially relevant for CPS, as they allow for different strategies in regards to the avoidance of detection. Capacity modulation can be helpful in scenarios where the characteristics of the cover channel vary. For example in physical processes the noise of sensor values might be different in transient states from steady states. On the other hand bandwidth modulation can be helpful to adjust the payload over time in regards to the physical process to avoid detection by a warden. Both strategies and especially in combination can help an attacker to optimize steganographic payload against detection probability.

In order to modulate the bandwidth in our approach, four **embedding strategies** $ES_i$ are applied to drive the *Cover Object Selection ($P_4$)*. For evaluation of this approach, we are using four different strategies, which employ different Embedding **Strategy Ratios** $ESr_i$, as shown in Table 1.

**Table 1: Embedding Strategies $ES_i$ and correlating Embedding Strategy Ratios $ES_{r_i}$**

| Strategy | Description | Strategy Ratio $ES_{r_i}$ |
|---|---|---|
| $ES_1$ | every suitable value | 1.0 |
| $ES_2$ | every second suitable | 0.5 |
| $ES_3$ | every third suitable value | 0.33 |
| $ES_4$ | pseudo-random method | variable |

$ES_1$ is the trivial case, when every suitable value (in the means of the cover model) is actually used for embedding or synchronisation data. In $ES_2$ we reduce the used values by 50% and only every second suitable value is going to be used. $ES_3$ uses every third suitable, therefore the strategy ratio is only 33% in relation to all suitable values. $ES_4$ is using a special method which uses a pseudo-random function to decide, whether a value is used or not. In order to be able to synchronise such pseudo-randomly inserted messages we are using a seed that is part of the stego key $K_S$.

As stated before, the idea of the embedding strategies is to modulate the bandwidth in the hope to render a detection less likely. The embedding **strategy ratio** $ES_{r_i}$ indicates how many of the sensor values, that are within the constraints of the cover model $CM_i$, are actually used for embedding. When also taking into values into account, which were not used for embedding due to the fact that there not apt for plausible embedding (e.g. due low noise), the embedding ratio $r$ is used: $r = \frac{|\bar{y}_{S_i}|}{|\hat{y}_{S_i}|+|\bar{y}_{S_i}|}$.

The **embedding ratio** $r$ indicates how many values were modified ($\bar{y}$) to embed a message in relation to all values (non-modified $\hat{y}$ due to $CM_i$ and $ES_i$ and modified $\bar{y}$). As the strategy ratio $ES_{r_i}$ defines how many values are actually used for embedding the hidden message, it is a direct parameter for the resulting embedding ratio $r$ and bandwidth. One of the major features of the pseudo-random method is that the strategy ratio $ES_{r_i}$ can be arbitrarily set by giving the function a probability with which is decided, whether a value is used or not. This method allows for fine-granular tuning of the embedding ratio $r$ and therefore is especially of interest in regards to make detection and suppression more difficult. This is even more embraced as the embedding seems to be random for an outside viewer. The embedding strategies are applied to both synchronisation sequences as well as sequences used for embedding the actual message. Figure 2 illustrates this at the example of the two sensors and embedding strategies.
The results of the application of these embedding strategies and their effect on the bandwidth are discussed in Section 4.7.

## 4.5 Embedding ($P_5$)

In $P_5$ the actual embedding of hidden bits for a given value takes place, whereas the Cover Model $CM_i$ of Process Step $P_1$ dictates if these are part of the synchronisation sequence or taken from the actual message (see Section 4.1). Either way, both the synchronisation sequences as well as the secret message get encrypted, encoded and split into embedding blocks and embedded in the target value (see Figure 3). In accordance to our attack scenario, where we assume an attacker wants to exfiltrate sensible information, we are generating a realistic hidden message with detailed information about the hardware and firmware of the target system. We are using the common Linux command `lshw` to generate a detailed hardware list and strip unnecessary symbols and line breaks. For our testing environment, this message is about 13.5 kilobytes in size. AES-ECB is currently used out of simplicity and can be exchanged by other modes, as our synchronisation methods are especially designed for this use case. As shown in Figure 3 the process of embedding can be divided in to five steps. In the first step the value, which is going to be modified in order to embed a part of the hidden message, is split in two parts: head and tail. The size of the parts is given by the *capacity* $cap_i$. The capacity is either calculated (and by that dynamically chosen) by using the cover model $CM_i$, in order to set a capacity in relation to the current noise, or a static capacity $cap_i$ is used which then is part of the stego key $K_S$ For the two sensors $S_1, S_2$, we tested capacities between 1 and 12 bit. Therefore, the algorithm splits the last four digits (tail) from the rest of the value (head). Those last four digits (tail) are then treated as an 16-bit integer. In the next step the tail gets transcoded to a binary string. From the encrypted message queue the next block in the size of the chosen capacity $cap_i$ is retrieved, which will then replace the last bits of the tail. In the third step this bit-block replaces the corresponding last bits of the original. After that, the new binary string gets encoded back to an integer and in the last step joined with the original.

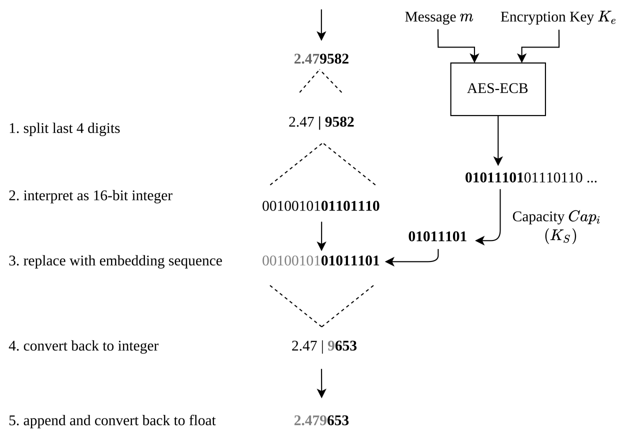

**Figure 3: The five-step process for embedding secret messages in the last digits of sensor values.**

## 4.6 Retrieval ($P_6$)

As described in our attack scenario in Section 3, the retrieval is done by a receiver who has access to the historian database. A common example would be an engineer accessing the database for monitoring or maintenance on the physical process. To be able to retrieve the message, the retriever has to have knowledge on (1) the algorithm, (2) which cover channels (sensors) the covert sender has access to and (3) the stego key $K_S$. As shown in Figure 1 the **Stego Key** $K_S$ is made out of six components: (1) Cover Channels (Selection Method) $\{C_i\}$, (2) Embedding Strategy $ES_i$, (3) (Embedding) Strategy Ratio $ES_{r_i}$, (4) Synchronisation Method $Sync_i$, (5) Capacity $Cap_i$ and (6) the Encryption Key $K_e$. In the context of our attack scenario we can assume the stego key to be static, e.g. as part of a supply chain attack. On a first glimpse the retrieval seems like the search for a needle in an haystack. However, the stego key $K_S$ is constructed in a way, to make this search successful: With the knowledge of the retriever which cover channels the covert sender has access to and which strategy is used to select cover channels $\{C_i\}$, the retriever knows where to search for hidden messages. With the knowledge of the algorithm and the synchronisation method $Sync_i$ the receiver can create a cover model. With that model and the knowledge on the used embedding strategy and its correlating strategy ratio $ES_i, ES_{r_i}$, the receiver can identify the synchronisation sequences, which contain information where the hidden messages are stored. In order to decode and decrypt these, the receiver needs to know the capacity $cap_i$ and the encryption key $K_e$.

## 4.7 Comparison of the Embedding Strategies

For the practical evaluation we are using the four embedding strategies $ES_i$ described in Section 4.4 for both sensors $S_1, S_2$ to evaluate the effect on the resulting bandwidth per sensor. In order to get comparable, yet realistic data, we are using a static capacity per cover object of 8 bit ($cap_{1,2} = 8$). As shown in Table 2 the results illustrate that the water flow sensor $S_1$ does not embark the same amount of potential bandwidth. From the water level sensor $S_2$ 53% of the measurements are supposed to be suitable (noise above 8 bit) in the context of the cover model, which is reflected in the **maximum embedding ratio** $r_{max}$ which describes how many of all measurements can be used for plausible embedding. In contrast to that, only 35% of the values of the water flow sensor $S_1$ are supposedly suitable for embedding ($r_{max} = 0.35$). This directly effects the maximum achievable bandwidth per sensor. Within the constraints of the cover model the maximum achievable bandwidth for the water level sensor $S_2$ is about 4.2 bit per second and 2.84 bit for the water flow sensor $S_1$ when using every suitable value for embedding ($ES_1$). To put this in perspective, the resulting bandwidth is set around 370kb (water level) and 245kb (water flow) per day. This might not seem to be a high bandwidth on a first glance, but in the context of the exfiltration attack scenario (which assumes a long-term projected attack) the bandwidth is comparable high ([10], [13]). For example, the hidden message we are using (which contains detailed hardware and firmware information about the target system) is only 13.5kb in size and therefore could be transmitted between 18 and 27 times depending on the sensor per day. Even when using an (embedding) strategy ratio of only 0.33 ($ES_r3 = 0.33$) on the lower performing water flow sensor $S_1$, in the course of a day the message could be transmitted six times (81kb). As the static 8 bit capacity is a conservative estimation, even more bandwidth might be achievable while also improving the stealthiness when using a dynamic capacity based on the local noise of the sensor.

## 5 DETECTION

In this section the concepts of our novel detection approaches are introduced. Furthermore, we will discuss how the approaches can be integrated in a real world ICS.

### 5.1 Detection Approaches

In order to detect the covert channel described in Section 4, we elaborate a novel detection approach based on [10]. The detection approach is based on the assumption that non-modified sensor data ($\hat{y}_{S_i}$) follows a rather uniform distribution on its last digits of sensor values on sequences with a supposedly short length (< 50 values) than steganographic sensor data ($\bar{y}_{S_i}$) as such aim for a uniform distribution during embedding in order to stay undetected. We assume that this steganographic embedding feature could result in data that is more uniformly distributed than unaltered data and leads to an detectable anomaly. Our detection approach is based on statistical pattern recognition and uses two classes of data for training and classification (non-modified sensor data ($\hat{y}_{S_i}$) and modified sensor data ($\bar{y}_{S_i}$)). Due to the fact, that a detection of an anomaly in a single sensor value is basically not possible because the steganographic embedding is hidden and is totally unobtrusive, the detector analyzes sequences of sensor values as introduced in [10]. The sensor value sequence $|Seq|$ indicates how many values ($y_{S_i}$) the detector analyzes for feature extraction and to create a sample (feature vector) for training or classification. The optimal

**Table 2: Embedding Strategy Comparison (sorted by Strategy Ration $ES_{r_i}$)**

| Strategy | | | Sensor | | | Results | |
|---|---|---|---|---|---|---|---|
| Strategy | Description | Strategy Ratio $ES_{r_i}$ | Sensor | Maximum Embedding Ratio $r_{max}$ | Embedding Ratio $r$ | Bandwidth bit/sec | kb/day |
| $ES_1$ | all | 1.00 | $S_1$ | 0.35 | 0.35 | 2.84 | 244.94 |
| $ES_1$ | all | 1.00 | $S_2$ | 0.53 | 0.53 | 4.28 | 369.75 |
| $ES_4$ | random | 0.75 | $S_1$ | 0.35 | 0.27 | 2.12 | 183.54 |
| $ES_4$ | random | 0.75 | $S_2$ | 0.53 | 0.40 | 3.21 | 276.98 |
| $ES_2$ | alternating | 0.50 | $S_1$ | 0.35 | 0.18 | 1.41 | 122.14 |
| $ES_2$ | alternating | 0.50 | $S_2$ | 0.53 | 0.27 | 2.14 | 184.88 |
| $ES_3$ | thirds | 0.33 | $S_1$ | 0.35 | 0.12 | 0.94 | 81.43 |
| $ES_3$ | thirds | 0.33 | $S_2$ | 0.53 | 0.18 | 1.42 | 122.81 |

size for the sensor value sequence $|Seq|$ has to be determined during evaluation by exploratory analyzing different length of $|Seq|$. We assume that the detector delivers the best performance with a sequence length between $|Seq| \geq 15$ and $|Seq| \leq 50$. The detector analyzes the last three digits (because of the 8-bit embedding) of multiple sensor values in a sequence (how many values depends on $|Seq|$ as mentioned before). In formalized terms it means that, the detector looks at the last three digits $x_1$, $x_2$ and $x_3$ of a sensor value $y_{S_i}$ where: $x_1 = y_{S_i}[length(y_{S_i}) - 3]$, $x_2 = y_{S_i}[length(y_{S_i}) - 2]$ and $x_3 = y_{S_i}[length(y_{S_i}) - 1]$. The approach stores all $x_1$,$x_2$ and $x_3$ for value sequence in three corresponding vectors $Vec\{x_1\}$, $Vec\{x_2\}$ and $Vec\{x_3\}$. For example: If $|Seq| = 3$, the detector analyses three sensor values $y_{S_i}$ in a sequence, e.g.: $y_{S_{i1}} = 2.456\mathbf{358}$, $y_{S_{i2}} = 421.2452\mathbf{514}$ and $y_{S_{i3}} = 234.23\mathbf{521}$. Thus, the resulting vectors for analysis are $Vec_{x1} = \{3, 5, 5\}$, $Vec_{x2} = \{5, 1, 2\}$ and $Vec_{x3} = \{8, 4, 1\}$ and have the length of $|Seq|$. This is only a short example to understand the basis of our approach, because the features of the approach are calculated based on these three vectors. As mentioned before, we assume a significant longer sequence size $|Seq|$ somewhere between $\geq 15$ and $\leq 50$ for an expedient detection. However, these three created vectors are analyzed by the approach hereafter. Initially, the percentage probability of occurrence for digits 0 to 9 is determined for every vector which results in ten percentage values for each vector. These 10 values for each of the three vectors are the basis for our feature extraction as in [10]. The standard deviation is calculated for every vector based on the ten calculated percentage values for the occurrence of a digit. It results in three extracted features $Std_{x1}$, $Std_{x2}$ and $Std_{x3}$. We determine these features based on our assumption, a more uniform distribution (as assumed for steganographic samples) of the values should result in a lower standard deviation. Additionally, we determine the variance $Var_{Std}$ and the average $Avg_{Std}$ of $Std_{x1}$, $Std_{x2}$ and $Std_{x3}$ because these values should be significantly differ for uniform and rather uniform distributions of $x_1$, $x_2$ and $x_3$. Furthermore, we extract the percentage maximum of the percentage values for the occurrence of a digit for every vector $Max_{x1}$, $Max_{x2}$ and $Max_{x3}$. We suppose that these values should be higher for non-steganographic samples because of the rather uniform distribution of the digits. This completes the feature extraction of our approach. All feature vectors are labeled for our statistical pattern recognition based detection approach. Thus our feature vector is 8-dimensional ($Std_{x1}$, $Std_{x2}$, $Std_{x3}$, $Var_{Std}$, $Avg_{Std}$, $Max_{x1}$, $Max_{x2}$ and $Max_{x3}$) with a label. The resulting feature vector represents a sample for training or classification. Based on the described 8-dimensional feature space, a two-class (non- steganographic data class and steganographic data class) J48 decision tree [19] is trained with WEKA3.9 [7]. We evaluated three other classifiers: logistic model tree (LMT), support vector machine (SVM) and multilayer perceptron (MLP). We use J48 decision tree because it performs with best accuracy and fastest runtime among the tested classifiers. The detector is in the following called $D_1$. Furthermore, we use WEKA to carry out a feature selection to see if the a subset of features may result in a better detection performance. Therefore, we use the *InfoGainAttributeEval* with its *Ranker* method with default parametrization to determine the information gain of a feature with respect to its class. The selection shows that $Std_{x3}$ is the feature with the clearly best separation precision. Due to this, we additionally train a second detector only with $Std_{x3}$ feature to train a J48 decision tree $D_2$. The J48 classifier is used in WEKA's default parametrization. Both approaches are trained with the same non-steganographic data (non-modified) and steganographic (modified) data, presented in Section 6.2.1. In our evaluation we will evaluate both detectors with same test data sets (presented in Section 6.2.2). We will determine and compare their performance in Section 6.3.

## 5.2 Integration of the detection approaches in ICS

The introduced detection approaches can be used as online or offline approaches. This means they can be integrated into a system to work live or be used with recorded data. If they are integrated in an live environment, the approach can practically make a classification in nearly real time. If the approaches work offline on recorded data, the detection can be slightly delayed in comparison to an online integration of the approaches. However, we see several options for both online and offline application at different positions in the reference architecture, e.g. to use online detectors in the network communication of PLCs and offline detectors directly on the historian. In order to assure that the detection has no influence on the real time capabilities and integrity of the system a decentralized integration of online-detectors seem reasonable. Here one scenario would be to place online detectors for each function unit or to use detectors for each PLC. Offline detectors might be especially of use when regularly crawling the historian database for potential hidden messages. The combination of both is especially interesting when using them to correlate events, thus making a detection more likely and easier to pinpoint infected devices.

## 6 EVALUATION OF DETECTION APPROACHES

In this section, we present our extensive evaluation of our previously introduced detection approaches. We describe evaluation goals, data and results in the following subsections.

### 6.1 Evaluation Goals

For our evaluation we define 3 goals to evaluate and compare our presented detection approaches $D_1$ and $D_2$. The evaluation goals can be summarized as follows:

- Goal 1: Determination of detection performance of $D_1$ & $D_2$ on different test data sets $TDS$ with different embedding strategies and ratios $r$.
- Goal 2: Comparison of their performance.
- Goal 3: Investigation of correlation between embedding ratio $r$ and detection performance.

The performance of the detection approaches is measured with well-known metrics. We use the True Positive Rate (TPR) for correct classified steganographic samples in a test data set and the True Negative Rate (TNR) for correct classified non-modified samples in test data. Additionally, we use the False Positive Rate (FPR) as complementary rate to TNR to describe triggered false alarms by the approach.

### 6.2 Evaluation Data

*6.2.1 Training Data.* To train our pattern recognition based detection approaches, we create a training data set $DS_{Training}$. Therefore, we use recorded authentic (non-modified) data ($\hat{y}_{S_i}$) and (modified) data with steganographic embedding ($\bar{y}_{S_i}$) from the two previously introduced sensors ($S_1$ and $S_2$). We use data from both sensors for training due to the expectation that a single detector will be able to detect anomalies for both sensors if it is trained with sensor data from both sensors. The steganographic training data includes only values with steganographic embedding. Thus, the training data has an embedding ratio $r = 1.0$. As mentioned before, $r$ indicates the ratio of modified values to all values in a data set ($r = \frac{|\bar{y}_{S_i}|}{|\hat{y}_{S_i}|+|\bar{y}_{S_i}|}$) and the resulting sample. $DS_{Training}$ contains 9529 values of non-modified sensor data $\hat{y}_{S_i}$ (4005 water flow samples $\hat{y}_{S_1}$, 5524 water level samples $\hat{y}_{S_2}$) and 7494 values with steganographic embedding $\bar{y}_S$(1688 flow samples $\bar{y}_{S_1}$, 5804 level samples $\bar{y}_{S_2}$). Thus, we have a small bias in training because we have slightly more non-steganographic values in $DS_{Training}$ because not every value is suitable for embedding, as mentioned before. However, this results in a different number of samples (remember: **samples** are **not values**, samples (or feature vectors) are calculated based on sensor values $Y_{S_i}$) for $DS_{Training}$, depending on the sequence size $|Seq|$ (see Section 5.1) which is used from the detection approach. Furthermore, we use a sliding window to analyze a data set with an overlapping of 50% to analyze the data more detailed than without a sliding window and to create more samples. This leads to a varying number of samples for a detector based on the sliding window (always 50% in this work) and the sequence size $|Seq|$ (visualized in Table 3). As mentioned before, the used sequence sizes for detection are determined by exploratory analyzing different sequence sizes. In the exploratory analysis, we evaluated our detectors with different $|Seq|$ in a 10-fold cross validation. We start with $|Seq| = 10$ and increase it step-wise by 10. Around $|Seq| = 20$ we determined the highest performance, we extend our exploratory analysis with small adjustments in this area of $|Seq|$ to find $|Seq|$ with the optimal performance. The analysis revealed that the optimal value sequence length for our detection approaches is $|Seq| = 22$. As a result we considered the following value sequences $|Seq| = \{10, 18, 19, 20, 21, 22, 23, 24, 30, 40, 50, 100\}$ in our evaluation.

**Table 3: Number of Samples for $DS_{training}$ for used sequence sizes $|Seq|$ ($|Seq|$ determined by exploratory analysis)**

| $\lvert Seq \rvert$ | Number of Samples based on non-modified Values $\hat{y}_{S_i}$ | Number of Samples based on modified Values $\bar{y}_{S_i}$ | Overall |
|---|---|---|---|
| 10 | 1905 | 1497 | 3402 |
| 18 | 1058 | 831 | 1889 |
| 19, 20 | 952 | 748 | 1700 |
| 21, 22 | 865 | 680 | 1545 |
| 23, 24 | 793 | 623 | 1416 |
| 30 | 634 | 498 | 1132 |
| 40 | 475 | 373 | 848 |
| 50 | 380 | 298 | 678 |
| 100 | 189 | 148 | 337 |

*6.2.2 Test Data.* To evaluate the performance of our elaborated detection approaches (presented in 5.1), we create 12 test data sets $TDS$, six for each sensor ($S_1$ and $S_2$). We summarize the test data sets in Table 4. We separate the sensor data for our evaluation to see if detectors, trained with water flow ($S_1$) and water level ($S_2$) samples, are able to detect anomalies for both sensors. We use the four different embedding strategies (introduced in 4.4) for both sensors which result in different embedding ratios $r$ to evaluate the detection performance in relation to the steganographic embedding ratio. Additionally, we use two test data sets with non-steganographic data only (one for each sensor) to see if the detector triggers false alarms (false positives). For $TDS_{S_1 r1.00}$ and $TDS_{S_2 r1.00}$ we refer to no embedding strategy because we simply exclude every sample in their initial data sets ($TDS_{S_1 r0.50}$ and $TDS_{S_2 r0.95}$) with no steganopgrahic embedding, which corresponds to a embedding ratio $r$ of 1.0. As mentioned before embedding ratio indicates how many values in a data set include steganographic embedding in a data set. For example: If the embedding ratio is 0.3 and the sequence size $|Seq| = 20$, six values in the analyzed sequence (to create a sample) include steganographic embedding ($\bar{y}_{S_i}$) and 14 values are non-modified ($\hat{y}_{S_i}$). Thus, a sample (feature vector) is a steganographic one as soon as one value contains steganographic embedding ($r > 0$) in the sequence. This means all steganographic test data sets with $r < 1.0$ ($TDS_{S_i r0.xx}$) contain sensor modified values with steganographic embedding ($\bar{y}_{S_i}$) as well as non-modified sensor values ($\hat{y}_{S_i}$). The embedding ratios varies for test data sets with steganographic embedding ($TDS_{S_i r_x}$) in comparison to descriptions in Table 2, because they are pre-processed for our evaluation. The pre-processing excludes long sequences ($|Seq| > 10$) with non-modified values (values that are not suitable for embedding) to avoid samples with no steganographic embedding in a steganographic data set because this would result in false labeled samples and biased evaluation results. With these different data sets (which include different embedding strategies and embedding ratios), we evaluate the impact of different steganographic embedding ratios induced by the different embedding strategies on the performance of detection. All in all, we have 2 $TDS$ with modified values only ($r = 1.0$; detection performance determined with TPR), 8 $TDS$ with mixed modified and non-modified values based on different embedding strategies ($0.0 > r < 1.0$; detection performance also determined with TPR, because we make sure that every calculated sample contains at least one modified value) and 2 $TDS$ with non-modified values only ($r = 0.0$, performance determined with TNR). The test data sets are analyzed by the detectors analogous to the training data set with the same sliding window of 50% and the same sequence sizes $|Seq|$. With the number of values of a data set the number of samples can be calculated based on used sequence size in consideration of the sliding window (50%).

**Table 4: Test data sets (independent from training) for the evaluation of detection approach**

| Name | Number of values $y_{S_i}$ | Embedding Ratio $r$ | Embedding Strategy | Type |
|---|---|---|---|---|
| $TDS_{S_1 r1.00}$ | 1279 | 1.00 | manual | $\bar{y}_{S_1}$ |
| $TDS_{S_1 r0.50}$ | 5924 | 0.50 | $ES_1$ | $\bar{y}_{S_1}$, $\hat{y}_{S_1}$ |
| $TDS_{S_1 r0.37}$ | 5924 | 0.37 | $ES_4$ | $\bar{y}_{S_1}$, $\hat{y}_{S_1}$ |
| $TDS_{S_1 r0.25}$ | 11785 | 0.25 | $ES_2$ | $\bar{y}_{S_1}$, $\hat{y}_{S_1}$ |
| $TDS_{S_1 r0.17}$ | 11584 | 0.17 | $ES_3$ | $\bar{y}_{S_1}$, $\hat{y}_{S_1}$ |
| $TDS_{S_1}$ | 2004 | 0.00 | - | $\hat{y}_{S_1}$ |
| $TDS_{S_2 r1.00}$ | 3038 | 1.00 | manual | $\bar{y}_{S_2}$ |
| $TDS_{S_2 r0.95}$ | 9294 | 0.95 | $ES_1$ | $\bar{y}_{S_2}$, $\hat{y}_{S_2}$ |
| $TDS_{S_2 r0.70}$ | 5924 | 0.70 | $ES_4$ | $\bar{y}_{S_2}$, $\hat{y}_{S_2}$ |
| $TDS_{S_2 r0.50}$ | 8954 | 0.50 | $ES_2$ | $\bar{y}_{S_2}$, $\hat{y}_{S_2}$ |
| $TDS_{S_2 r0.33}$ | 9067 | 0.33 | $ES_3$ | $\bar{y}_{S_2}$, $\hat{y}_{S_2}$ |
| $TDS_{S_2}$ | 2761 | 0.00 | - | $\hat{y}_{S_2}$ |

## 6.3 Evaluation Results

In this section we discuss the evaluation results for our detectors $D_1$ and $D_2$ trained with presented data in 6.2.1 and tested with data presented in 6.2.2. This section is structured according to the evaluation goals 1, 2 and 3 presented in Section 6.1.

*6.3.1 Performance of Detector $D_1$ and $D_2$ (Goal 1).* The detection performance results for $D_1$ and $D_2$ are presented in Table 5. Both Detector $D_1$ and $D_2$ can reach their best performance results with a value sequence length of $|Seq| = 22$. The detection performance varies strongly for different $|Seq|$, even for small variations of $|Seq|$ (see results for $|Seq| = 23$ or $|Seq| = 21$). We have not found a comprehensive reason for this, yet. However, for $|Seq| = 22$ the feature spaces $D_1$ and $D_2$ deliver the best separation precision between the two classes of data. As assumed before, the optimal size or length for $|Seq|$ has to be quite low that the features have a separation precision because the steganographic embedding is warden compliant (see [13]) for longer value sequences (we assume $|Seq| > 250$ by initial tests). We evaluate the performance of detectors only for $|Seq| < 100$ because we can observe a dropping detection performance and a less accurate classification of non-modified samples for $|Seq| \geq 50$. **Detector $D_2$ is able to detect 87.8% (TPR) of steganographic samples (for $TDS_{S_1 r1.00}$) at a FPR of 0% (TNR = 100% for both test datasets with non-modified data).** $D_1$ delivers a comparable detection rate with up to 81.8% of correct classified steganographic samples but with a TPR of 93.9% for $TDS_{S_1}$ and 94.8% for $TDS_{S_2}$. The detection performance of both detectors drops significantly when the embedding ratio $r$ decreases (e.g. 28.7% correct classified samples for $D_2$ and 30.3% $D_1$ when the embedding ratio $r = 0.17$). We will compare the detection performance of $D_1$ and $D_2$ in 7.3.2. All in all, both detectors deliver promising results, but with a significant variance between different $|Seq|$. Furthermore, we can observe comparable detection results between $S_1$ and $S_2$ test data sets. We conclude that the detection has a similar performance for both types of data with a slightly better detection in $S_1$ data sets because both detectors have slightly better TPR on water flow ($S_2$) data sets. We assume, that this is caused by the training data. Furthermore, where we previously assumed that the embedding strategy is rather relevant for our detection, we can now see that the **more significant parameter** for detection is the **embedding ratio** $r$. We analyze the detection performance in context of the embedding ratio in detail in 7.3.3.

*6.3.2 Detector $D_1$ vs $D_2$ (Goal 2).* For the comparison of $D_1$ and $D_2$, we visualize the performance of the detectors for each test data set with $|Seq| = 22$ in Figure 4. We can state that the performance of $D_2$ for non-modified samples is perfect (with $|Seq| = \{19, 20, 21, 22, 23, 40\}$) in our evaluation, not a single non-modified sample is classified as a steganographic-one, which means no false alarms are triggered by $D_2$ (FPR = 0%). The performance of $D_1$ for non-modified test data sets is significant lower, it delivers a false positive rate (FPR) of 6.1% (93.9% TNR) for $TDS_{S_1}$ and a FPR of 5.2% (94.8% TNR) for $TDS_{S_2}$ for $|Seq| = 22$. Thus, the **performance on non-modified data is significantly better with our single-feature detector $D_2$**. In terms of detection performance both detectors deliver their best performance with a $|Seq| = 22$. The results are comparable except for $TDS_{S_1 r1.00}$. For $TDS_{S_1 r1.00}$ detector $D_2$ delivers a significant higher performance than $D_1$ with a 87.8% vs. 81.7%. For other test data sets with steganographic data and a embedding ratio $r \geq 0.35$, $D_2$ performs slightly better than $D_1$. For a embedding ratio $r < 0.35$ $D_1$ delivers a better TPR than $D_2$. Overall, we postulate that the single feature detector $D_2$ **delivers better performances in our evaluation** but for a **real world application, we recommend to use** $D_1$ first, because we assume a more robust detection with $D_1$ on real world application with more diverse training and testing. The optimal value sequence length $|Seq|$ has to be re-determine for every field of application and its training and test sensor data. Additionally, $|Seq|$ can significantly vary because of different embedding strategies and other resulting anomalies.

*6.3.3 Embedding Ratio $r$ vs. Detection Performance (Goal 3).* During our evaluation, we observe an correlation of the steganographic embedding ratio $r$ and the detection performance. Obviously, we can state that higher embedding ratios result in a better detection performances (high TPR) and a decreasing performance when the

**Table 5: TNR and TPR for Detector $D_2$ and $D_1$ for different $|Seq|$ on test data sets (FPR complementary to TNR for $TDS_{S_1}$ and $TDS_{S_2}$).**

| $\|Seq\|$ | $TDS_{S_1}$ | | $TDS_{S_1}r1.00$ | | $TDS_{S_1}r0.50$ | | $TDS_{S_1}r0.37$ | | $TDS_{S_1}r0.25$ | | $TDS_{S_1}r0.17$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TNR($D_2$) | TNR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) |
| 10 | 99.1 | 95.2 | 41.7 | 46.1 | 28.9 | 31.2 | 28.1 | 31.7 | 19.1 | 22.1 | 13.9 | 18.0 |
| 18 | 99.5 | 91.9 | 68.8 | 70.1 | 49.8 | 54.0 | 48.3 | 50.5 | 33.9 | 38.1 | 24.3 | 28.8 |
| 19 | 100 | 85.4 | 54.0 | 43.7 | 38.7 | 30.5 | 43.3 | 34.2 | 30.8 | 28.5 | 21.1 | 21.6 |
| 20 | 100 | 93.5 | 78.6 | 78.6 | 57.2 | 56.7 | 53.8 | 55.3 | 39.1 | 39.6 | 26.7 | 28.3 |
| 21 | 100 | 84.5 | 61.7 | 59.1 | 51.2 | 50.3 | 48.8 | 49.9 | 35.4 | 39.9 | 25.0 | 31.9 |
| **22** | **100** | **93.9** | **87.8** | **81.7** | **61.6** | **61.6** | **56.4** | **55.9** | **39.3** | **40.1** | **28.7** | **30.3** |
| 23 | 100 | 85.5 | 68.6 | 64.8 | 27.4 | 36.6 | 18.7 | 30.9 | 10.4 | 25.7 | 6.4 | 21.4 |
| 24 | 99.4 | 70.3 | 67.6 | 48.6 | 53.4 | 40.2 | 53.9 | 44.9 | 39.8 | 41.7 | 28.9 | 36.1 |
| 30 | 82.6 | 70.5 | 61.9 | 50.0 | 47.1 | 43.8 | 48.6 | 42.7 | 39.3 | 39.4 | 34.5 | 36.4 |
| 40 | 100 | 64.6 | 79.0 | 62.9 | 59.3 | 45.8 | 54.2 | 47.5 | 39.1 | 42.2 | 28.5 | 40.8 |
| 50 | 74.7 | 68.4 | 46.0 | 52.0 | 29.8 | 43.0 | 34.0 | 43.4 | 33.2 | 42.3 | 32.5 | 36.4 |
| 100 | 28.2 | 82.1 | 87.5 | 62.5 | 81.2 | 45.3 | 81.2 | 35.9 | 77.4 | 29.9 | 75.1 | 32.6 |

| $\|Seq\|$ | $TDS_{S_2}$ | | $TDS_{S_2}r1.00$ | | $TDS_{S_2}r0.95$ | | $TDS_{S_2}r0.70$ | | $TDS_{S_2}r0.50$ | | $TDS_{S_2}r0.33$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TNR($D_2$) | TNR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) | TPR($D_2$) | TPR($D_1$) |
| 10 | 99.5 | 93.5 | 42.7 | 45,7 | 43.0 | 45.7 | 36.2 | 38.7 | 31.2 | 33.4 | 21.1 | 23.1 |
| 18 | 100 | 94.4 | 62.2 | 66.1 | 62.2 | 68.9 | 58.5 | 59.8 | 51.5 | 54. | 35.6 | 38.8 |
| 19 | 100 | 89.1 | 54.3 | 43.7 | 51.9 | 41.5 | 47.0 | 39.5 | 40.8 | 29.6 | 29.5 | 20.7 |
| 20 | 100 | 96.0 | 77.5 | 79.5 | 73.7 | 74.7 | 66.9 | 65.7 | 55.6 | 55.6 | 39.3 | 39.4 |
| 21 | 100 | 85.6 | 62.9 | 65.8 | 59.3 | 65.8 | 54.5 | 56.5 | 47.0 | 49.0 | 34.1 | 38.8 |
| **22** | **100** | **94.8** | **82.5** | **81.8** | **83.9** | **81.6** | **71.9** | **70.1** | **56.9** | **55.5** | **40.6** | **42.3** |
| 23 | 100 | 87.3 | 62.7 | 67.5 | 62.6 | 63.0 | 46.3 | 47.5 | 29.7 | 36.5 | 15.5 | 25.9 |
| 24 | 100 | 77.7 | 64.3 | 57.9 | 59.9 | 55.4 | 57.6 | 46.4 | 48.5 | 42.1 | 36.2 | 34.7 |
| 30 | 86.3 | 71.6 | 59.7 | 54.2 | 51.6 | 52.6 | 46.7 | 47.5 | 37.6 | 39.3 | 29.2 | 37.1 |
| 40 | 100 | 75.2 | 66.7 | 62.0 | 62.2 | 59.0 | 55.6 | 50.1 | 43.9 | 42.4 | 32.5 | 38.3 |
| 50 | 86.2 | 77.1 | 43.0 | 56.0 | 35.4 | 56.8 | 27.4 | 47.6 | 22.4 | 37.5 | 16.9 | 34.9 |
| 100 | 48.1 | 92.6 | 86.4 | 67.0 | 77.2 | 62.0 | 70.9 | 46.6 | 60.7 | 32.0 | 51.7 | 31.7 |

embedding ratio falls down. We achieve a detection performance of up to 87.8% when every value in a data set contains steganographic embedding ($r$=1.0) with a value sequence length $|Seq = 22|$. But, the **detection performance is more robust than expected, it decreases not as significant as the embedding ratio** $r$. If the embedding ratio drops for example to 25% ($TDS_{S_1}r0.25$) the detection performance is still around 40% TPR with $D_1$ and $|Seq| = 22$. It is interesting, that the detector $D_2$ is able to detect up to 28.7% of steganographic samples with $|Seq| = 22$ on the water-flow data set $TDS_{S_1}r0.17$ with a, embedding ratio of only $r = 0.17$ and a FPR = 0%. This means the detectors are able to detect (with $|Seq| = 22$) nearly one out of three samples correct, even if only 3 or 4 (in average 3.74) out of 22 values analyzed in a sample are modified with steganographic embedding. With sacrificing a high TNR (results in higher FPR), also higher detection performances of up to 40.8% for $D_1$ with $|Seq| = 40$ can be achieved for data sets with low embedding ratio $r = 0.17$ ($TDS_{S_1}r0.17$).

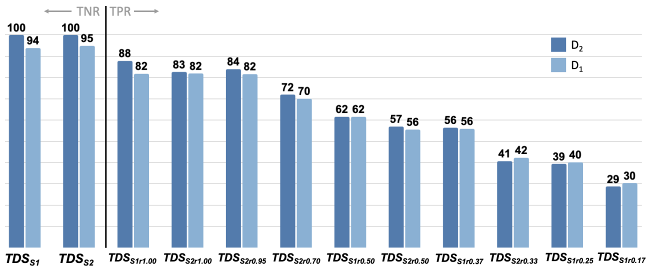

**Figure 4: Performance of detectors $D_2$ and $D_1$ with $|Seq| = 22$ for all test data sets sorted by embedding ratio**

## 7 CONCLUSION & FUTURE WORK

At the example of two types of sensors commonly found in Cyber Physical Systems (CPS), we present a network covert channel aiming at exfiltrating sensible information of Industrial Control Systems (ICS). Leveraging the circumstance that in such environments process data is usually stored persistently in historian databases, this channel is based on a protocol-agnostic approach, using long-term persistent asynchronous communication. By using different embedding strategies this approach is bandwidth adaptive, aimed to render detection by a warden less likely. Our experimental evaluation shows the feasibility of this approach for the selected attack scenario. In this first evaluation, using the proposed embedding strategies a bandwidth between 80kb and 370kb per day (depending on the sensor) could be achieved. Given a targeted long-term scaled attack scenario this covert channel seems to be a viable option for data exfiltration over a longer period of time. In the process of designing covert channels for CPS several challenges arise, especially in regards to cover channel selection, sender receiver synchronisation and retrieval. Our approach addresses these by proposing different methods. In the context of this specific covert channel we describe the general concepts of *recaching*, the difference of asynchronous, synchronous and ephemeral channels as well as the temporal aspect in from of *levels of persistence*. These concepts apply to covert channels in Cyber Physical Systems but might be applicable as well to other domains of Information Hiding. Additionally, we introduce two novel detection approaches $D_1$ and $D_2$ based on [10]. These detection approaches work with value sequences $|Seq|$ which are used to create a sample for training or classification. The optimal size or length for $|Seq|$ has to be determined by an exploratory analysis. $D_2$ achieves a TPR of 87.8% by a FPR of 0.0% for value sequence $|Seq| = 22$ for an embedding ratio of 100%. The performance of the approaches varies strongly for different $|Seq|$ (even for small adjustments) and decreases for $|Seq| > 50$. If the embedding ratio decreases the detection performance decreases significantly. With an embedding ratio of $r = 0.17$ $D_2$ is able to detect still 28.9% of steganographic samples (TPR) with 0% FPR and $|Seq| = 24$. In comparison, our single feature detector $D_2$ delivers better results in our evaluation than $D_1$ for embedding ratios < 32% with $|Seq| = 22$, $D_1$ has a slightly better TPR for embedding ratios ≥ 32%. The detection delivers slightly better results for water-flow sensor data sets than for water-level data sets, we assume this is caused by training. While the results for the proposed covert channel look promising, the presented methods shall be further analysed and investigated in the future. One major aspect for further investigation is the robustness of the presented covert channel in regards to its communication flow along several intermediate nodes, especially ones which employ protocol (or data-) conversions. Further improvements are planned for the cover model and noise calculation as well as testing a dynamic capacity selection based on the current noise of the sensor and process. The idea here is to fully automate the processes $P_1$ to $P_4$ Cover Model Generation, Cover Channel Selection, Cover Model Application and Cover Object Selection. Furthermore, the applicability of the presented approach on other types of sensors is investigated in the future as well. Another aspect to be tested in the context of this approach is *tactical deception* (e.g. [23]), which can be seen as a technique of anti-forensics in IT-Security.

## ACKNOWLEDGMENTS

The presented work is funded by the German Federal Ministry of Economic Affairs and Energy (BMWi, project no. 1501589A and 1501589C) in the framework of the German reactor safety research program. The authors thank all project partners, especially Robert Altschaffel, Jonas Hielscher and Mario Hildebrandt as well as the reviewers for their helpful comments.

## REFERENCES

[1] 2021. Deep Dive into the Solorigate Second-Stage Activation: From SUNBURST to TEARDROP and Raindrop. https://www.microsoft.com/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop/.

[2] Chuadhry Mujeeb Ahmed, Aditya Mathur, and Martin Ochoa. 2017. NoiSense: Detecting Data Integrity Attacks on Sensor Measurements Using Hardware Based Fingerprints. *arXiv:1712.01598 [cs]* (Dec. 2017). arXiv:1712.01598 [cs]

[3] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. 2017. WADI: A Water Distribution Testbed for Research in the Design of Secure Cyber Physical Systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWATER '17)*. Association for Computing Machinery, New York, NY, USA, 25–28. https://doi.org/10.1145/3055366.3055375

[4] Chuadhry Mujeeb Ahmed, Jianying Zhou, and Aditya P. Mathur. 2018. Noise Matters: Using Sensor and Process Noise Fingerprint to Detect Stealthy Cyber Attacks and Authenticate Sensors in CPS. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*. Association for Computing Machinery, New York, NY, USA, 566–581. https://doi.org/10.1145/3274694.3274748

[5] S. D. Antón, Michael Gundall, Daniel Fraunholz, and H. Schotten. 2019. Implementing SCADA Scenarios and Introducing Attacks to Obtain Training Data for Intrusion Detection Methods. *ArXiv* (2019).

[6] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. 2017. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In *Critical Information Infrastructures Security (Lecture Notes in Computer Science)*, Grigore Havarneanu, Roberto Setola, Hypatia Nassopoulos, and Stephen Wolthusen (Eds.). Springer International Publishing, Cham, 88–99. https://doi.org/10.1007/978-3-319-71368-7_8

[7] M. Hall. 2009. The WEKA data mining software: An update. *In SIGKDD Explorations* (2009).

[8] Theodore G. Handel and Maxwell T. Sandford. 1996. Hiding data in the OSI network model. In *Information Hiding*, Ross Anderson (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 23–38.

[9] Amir Herzberg and Yehonatan Kfir. 2019. The Chatty-Sensor: A Provably-Covert Channel in Cyber Physical Systems. In *Proceedings of the 35th Annual Computer Security Applications Conference (ACSAC '19)*. Association for Computing Machinery, New York, NY, USA, 638–649. https://doi.org/10.1145/3359789.3359794

[10] Mario Hildebrandt, Kevin Lamshöft, Jana Dittmann, Tom Neubert, and Claus Vielhauer. 2020. Information Hiding in Industrial Control Systems: An OPC UA Based Supply Chain Attack and Its Detection. In *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*. ACM, Denver CO USA, 115–120. https://doi.org/10.1145/3369412.3395068

[11] Andrew D. Ker. 2007. Batch Steganography and Pooled Steganalysis. In *Information Hiding (Lecture Notes in Computer Science)*, Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee (Eds.). Springer, Berlin, Heidelberg, 265–281. https://doi.org/10.1007/978-3-540-74124-4_18

[12] Marina Krotofil, Jason Larsen, and Dieter Gollmann. 2015. The Process Matters: Ensuring Data Veracity in Cyber-Physical Systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/2714576.2714599

[13] Kevin Lamshöft and Jana Dittmann. 2020. Assessment of Hidden Channel Attacks: Targetting Modbus/TCP. *IFAC-PapersOnLine* 53, 2 (2020), 11100–11107. https://doi.org/10.1016/j.ifacol.2020.12.258 21th IFAC World Congress.

[14] Kevin Lamshöft, Tom Neubert, Mathias Lange, Robert Altschaffel, Mario Hildebrandt, Yongjian Ding, claus Vielhauer, and Jana Dittmann. 2020. Novel Challenges for Anomaly Detection in I&C Networks: Strategic Preparation for the Advent of Information Hiding Based Attacks. *ATW Journal* 65 (Oct. 2020), 504–508.

[15] Antoine Lemay and José M Fernandez. [n.d.]. Providing SCADA Network Data Sets for Intrusion Detection Research. ([n. d.]), 8.

[16] The Mitre Foundation. [n.d.]. Attackics. https://collaborate.mitre.org/attackics/.

[17] Steven J. Murdoch and Stephen Lewis. 2005. Embedding Covert Channels into TCP/IP. In *Information Hiding*, Mauro Barni, Jordi Herrera-Joancomartí, Stefan Katzenbeisser, and Fernando Pérez-González (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 247–261.

[18] National Institute of Standards and Technology (NIST). 1983. *Trusted Computer System Evaluation Criteria*. National Institute of Standards and Technology.

[19] R. Quinlan. [n.d.]. C4.5: Programs for Machine Learning. *1993 Morgan Kaufmann Publishers, Inc.; ISBN 1-55860-238-0;* ([n. d.]).

[20] Pascal Schöttle and Rainer Böhme. 2013. A Game-Theoretic Approach to Content-Adaptive Steganography. In *Information Hiding (Lecture Notes in Computer Science)*, Matthias Kirchner and Dipak Ghosal (Eds.). Springer, Berlin, Heidelberg, 125–141. https://doi.org/10.1007/978-3-642-36373-3_9

[21] S. H. Sellke, C. . Wang, S. Bagchi, and N. Shroff. 2009. TCP/IP Timing Channels: Theory to Implementation. In *IEEE INFOCOM 2009*. 2204–2212. https://doi.org/10.1109/INFCOM.2009.5062145

[22] Pukhraj Singh. 2005. Whispers on the Wire-Network Based Covert Channels. In *Proceedings of the Symposium on Security for Asia Network (SyScAN'05)*. Bangkok, Thailand.

[23] Michael A. Steele, Sylvia L. Halkin, Peter D. Smallwood, Thomas J. McKenna, Katerina Mitsopoulos, and Matthew Beam. 2008. Cache Protection Strategies of a Scatter-Hoarding Rodent: Do Tree Squirrels Engage in Behavioural Deception? *Animal Behaviour* 75, 2 (Feb. 2008), 705–714. https://doi.org/10.1016/j.anbehav.2007.07.026

[24] Aleksandar Velinov, Aleksandra Mileva, Steffen Wendzel, and Wojciech Mazurczyk. 2019. Covert Channels in the MQTT-Based Internet of Things. *IEEE Access* 7 (Nov. 2019), 161899–161915. https://doi.org/10.1109/ACCESS.2019.2951425

[25] Steffen Wendzel, Worms University of Applied Sciences, Germany, Wojciech Mazurczyk, Fraunhofer FKIE, Germany, Georg Haas, and Warsaw University of Technology, Poland. 2017. Steganography for Cyber-physicalSystems. *Journal of Cyber Security and Mobility* 6, 2 (2017), 105–126. https://doi.org/10.13052/jcsm2245-1439.621

[26] Theodore J. Williams. 1994. The Purdue Enterprise Reference Architecture. *Computers in Industry* 24, 2 (Sept. 1994), 141–158. https://doi.org/10.1016/0166-3615(94)90017-5