Accurate Automata-Based Detection of Cyber Threats in Smart Grid Communication

Vojtěch Havlena, Petr Matoušek, Ondřej Ryšavý, and Lukáš Holík

***Abstract*—Several industry sectors, including critical infrastructure, have experienced severe cyber attacks against their Industrial Control Systems (ICS) due to the malware that masqueraded itself as a legitimate ICS process and communicated with valid ICS messages. Such behavior is difficult to detect by standard techniques. Intrusion Detection Systems (IDS) usually filter illegitimate communication using pre-defined patterns while statistical-based Anomaly Detection Systems (ADS) mostly observe selected attributes of transmitted packets without deeper analysis of ICS messages. We propose a new detection approach based on Deterministic Probabilistic Automata (DPAs) that capture the intended semantics of the ICS message exchange. The method models normal ICS message sequences using a set of DPAs representing expected traffic patterns. Then the detection system applies reasoning about the model to reveal a malicious activity in the ICS traffic expressed by unexpected ICS messages. In this paper, we significantly improve the performance of the automata-based detection method and reduce its false-positive rate. We also present a technique that produces additional details about detected anomalies, which is important for real-world deployment. The approach is demonstrated on IEC 104 or MMS communication from different ICS systems.**

***Index Terms*—Smart grid, cyber security, anomaly detection, probabilistic automata, network flows, MITRE ATT&CK.**

Nomenclature

Abbreviations

AD	Anomaly Detection
ADS	Anomaly Detection System
APT	Advanced Persistent Threats
ASDU	Application Service Data Unit
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
CoT	Cause of Transmission
DLMS	Device Language Message Specification protocol
DNP3	Distributed Network Protocol 3
DoS	Denial of Service attack
DPA	Deterministic Probabilistic Automaton
FDIA	False Data Injection Attack
FPR	False Positive Rate
GOOSE	Generic Object Oriented Substation Events
HMI	Human-Machine Interface
ICS	Industrial Control System
IDS	Intrusion Detection System
IEC 104	International Electrotechnical Commission (IEC) 104 protocol, standard IEC 60870-5-104
IED	Intelligent Electronic Device
IPFIX	IP Flow Information Export protocol
MITM	Man in the Middle attack
MITRE	Massachusetts Institute of Technology Research & Engineering, a non-profit organization
MMS	Manufacturing Message Specification protocol
Modbus	Modicon Communication Bus protocol
OBIS	Object Identification System
PCAP	Packet Capturing data format
PDU	Protocol Data Unit
PLC	Programmable Logical Controller
RTU	Remote Terminal Unit

Algebraic Symbols

$\mathcal{A}$	Deterministic Probabilistic Automaton (DPA)
$\mathcal{A}_p$	a DPA corresponding to a pair of devices p
$\mathbb{A}$	set of probabilistic automata
$\mathbb{A}_p$	set of probabilistic automata corresponding to a pair of devices p
$\mathcal{C}_p$	multiset of conversations between a pair of devices p
$\mathcal{D}$	all pairs of devices communicating within the network
ϵ	model reduction threshold
$\mathcal{G}$	product automaton
$\mathcal{L}$	language (set of strings)
$\mathcal{P}_{\mathcal{A}}$	probabilistic distribution generated by DPA $\mathcal{A}$
Σ	alphabet, i.e., a set of characters
θ	anomaly threshold
p	pair of devices, $p \in \mathcal{D}$
u, v, w	strings/conversations

I. Introduction

SMART grid communication includes control and monitoring transmissions that are exchanged between Intelligent Electronic Devices (IEDs), Human-Machine Interfaces (HMIs), control stations, and gateways. Connected devices

Manuscript received 16 March 2022; revised 21 June 2022 and 18 August 2022; accepted 19 October 2022. Date of publication 25 October 2022; date of current version 21 April 2023. This work was supported by the Ministry of Interior of the Czech Republic through the Research Project "Security Monitoring of ICS Communication in the Smart Grid (Bonnet)," 2019–2022, under Grant VI20192022138. Paper no. TSG-00348-2022. *(Corresponding author: Petr Matoušek.)*

The authors are with the Faculty of Information Technology, Brno University of Technology, 601 90 Brno, Czech Republic (e-mail: matousp@fit.vutbr.cz).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2022.3216726.

Digital Object Identifier 10.1109/TSG.2022.3216726

1949-3053 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

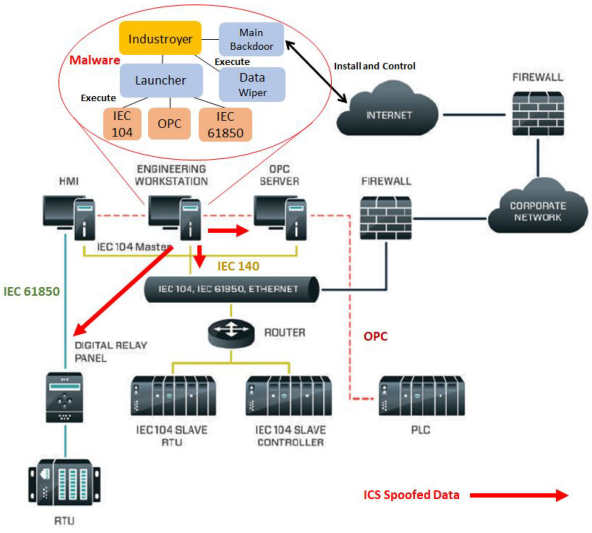


Fig. 1. Industroyer attack on power grid in 2016 [8].

typically communicate using standardized ICS protocols like IEC 104, MMS, GOOSE, DNP3, or DLMS [1]. The communication is often not secured which makes it an easy target for cyber attacks. Cyber security of industrial systems, including smart grids, has thus become a huge challenge due to devastating attacks on critical infrastructure around the world [2], [3], [4], [5]. Notable cases include disruption of Ukrainian energy distribution by malware BlackEnergy3 in 2015 and Industroyer/CrashOverride in 2016 [6], [7], [8], disconnection of safety instrumented system by malware Triton in 2017 and 2019 [9], [10], enumeration of Open Platform Communication servers by cyber espionage malware Dragonfly/Havex [11], [12], or the PLC-Blaster malware attacking Siemens S7 Programmable Logical Controllers (PLCs) [13].

Such attacks were driven by malware installed on an internal device or control station infected by social engineering techniques, supply chain compromise, or replication through removable media.[1] An ICS-capable malware usually employs industrial communications to discover ICS network resources, requests execution of unauthorized commands, collects sensitive data, or even manipulates ICS processes, see Fig. 1, that is not easy to detect using traditional techniques.

Widely deployed Intrusion Detection Systems (IDS) or firewalls detect anomalies (i) by checking protocol headers and applying white lists or black lists filtering based on observed values, or (ii) by searching individual ICS packet content for known patterns. Statistical-based Anomaly Detection Systems (ADS) compute quantitative characteristics of ICS traffic and create a probabilistic model of the normal behavior. Significant deviations in the observed characteristics from the model are then reported. However, malware often sends crafted but otherwise valid ICS messages which are difficult to distinguish from legitimate communication. Also, an attacker can adjust communication in such a way that it looks normal in terms of statistical properties by modifying the timing of transmitted messages. Detection of such attacks requires techniques that are sensitive both to the *content* of ICS messages and the *context* of the entire communication.

We have laid the foundation of a *content and context sensitive detection* in our previous work [14], [15]. Characteristics of a normal traffic are learned from long-term traffic samples in the form of deterministic probabilistic automata (DPAs) and then compared against DPAs synthesised from short-term windows of the traffic currently under scrutiny. The DPA captures probabilistic distribution of *conversations* among industrial devices. Typical conversations consist of a small number of commands related to a specific-purpose device and arranged in some of a few simple regular patterns [16], [17], [18], [19]. Our previous work [15] demonstrates that the DPA-based approach is feasible and is able to detect various network attacks on ICS systems.

A. *Scope and Motivation*

The scope of this paper is to present an automata-based anomaly detection method that models ICS communication and is suitable for practical deployment in a smart grid environment. The ICS traffic is modelled using Deterministic Probabilistic Automata (DPAs) [20] that capture ICS conversations together with their relative occurrences in the overall communication. This approach relies on network monitoring and requires information extracted from industrial protocols. In particular, packet headers containing message types and other attributes are processed. We do not deal with a payload; thus, it is not necessary to interpret system-specific values in our model. Even without knowing the payload, it is possible to detect various types of attacks, e.g., injection attacks, scanning attacks, switching attacks, etc., as demonstrated in the paper.

Two major issues related to the practical deployment of industrial anomaly detection systems are addressed in this work: the higher rate of false positives and interpretation of detection results. Our main motivation is to provide a solution that precisely detects anomalies, decreases the number of false positives, and gives an operator additional hints on how to easily interpret raised alarms.

Our original approach [15] suffered from a large number of *false positives* that—depending on the size of the system—may be hundreds or even thousands per day. False positives can hardly be entirely removed, but their number must be kept low to be acceptable in practice. The *anomaly reports are often difficult to interpret*, because the AD methods usually indicate a numerical distance of the analysed event from the normal one. The operator then have to manually inspect anomalous communication consisting of thousands of messages, thus, the analysis relies on his/her expert knowledge and experience. These two aspects put excessive demands on the operator and would likely render the method practically unattractive. In this paper, we provide a solution for both of these issues.

B. *Contribution*

This paper presents two main contributions: (i) elimination of false positives and (ii) providing diagnostic traces,

[1] See MITRE ATT&CK Tactics at `https://attack.mitre.org/tactics/TA0001/` [Jan 2022].

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

accompanied by two additional results that include (iii) extension of our original detection approach tailored for the protocol IEC 104 to other ICS protocols, such as MMS, and (iv) mapping of the capabilities of our detection method to common MITRE adversarial tactics [21].

Concerning contribution (i): False positives and general imprecision of the detection are caused by the fact that the traffic is variable over time, depending on factors such as time of the day, a particular task given to the industrial system, etc. Since all the variability is summarised in a *single DPA* model, the particular traffic windows taken under different conditions, even though perfectly normal, would likely differ from the learned summary DPA, resulting in the large false positive rate. Our solution has two parts: (i) in the learning phase, we *split the large learning traffic sample* into multiple small parts and learn a separate DPA from each part; (ii) in the detection phase, the traffic window under scrutiny is *compared against the most similar* learned DPA. This solution dramatically improves precision of the method and eliminates almost all false positives.

However, a naïve implementation of this techniques has a practical limitation. During the learning we obtain hundreds of DPAs representing a normal traffic. Comparing the testing traffic against each of them in the detection phase, in order to find the most similar DPA, is very expensive. To address this issue, we came up with a method for reducing the number of learned DPAs. To this end, we represent a *cluster* of similar DPAs by a single *representative*. The traffic under scrutiny is then compared only against the representatives of clusters. Our method allows to define a level of DPA similarity which in turn gives a control over the detection error caused by clustering. Consequently, we can prevent any notable deterioration of the detection precision. The representatives can be *pre-computed* and hence there is no additional cost during real-time detection. The clustering reduces the number of DPAs from *hundreds to small units* that can be handled in run-time perfectly well.

Concerning contribution (ii): To facilitate the analysis of the anomaly reports, the proposed improvement generates *diagnostic traces*. Namely, we demonstrate how to find examples of conversations from the tested traffic that was *not expected* to occur according to the normal traffic model, and how to synthesise examples of conversations that were *expected to occur but did not*. The work of the operator hence reduces from a blind analysis of the entire traffic window into the inspection of a few samples of conversations.

The proposed improvements are non-trivial and indeed turn the proof of concept published in [15] into a practical method (currently being implemented into a commercial solution). We present an empirical evidence of the effectiveness of the proposed techniques based on a prototype implementation. We also provide testing datasets.

C. Structure of the Paper

Section II reviews related work and presents used nomenclature. Section III stands for the paper's core and explains the detection method. After recalling the basics of DPA-based anomaly detection, it includes our contributions: multiple-model detection, model reduction, and generating diagnostic traces. Section IV presents the experimental evaluation of the method using implemented prototype. Based on the results from experiments, the comparison to other work is presented in Section V. Section VI concludes the paper by summarizing results and providing notes on real-world deployments.

II. Related Work

According to Rakas et al. [22], anomaly detection systems are statistical-based (univariate, multivariate, time series), knowledge-based (finite automata, expert systems), or machine learning-based (applying Bayesian networks, Markov models, neural network, fuzzy logic). Anomaly detection methods mainly work with statistical features obtained from ICS traffic, or they build a communication model using attributes extracted from IP or TCP headers.

Many recent works related to smart grid security focus on detecting false data injection attacks (FDIA) [23], [24], [25] where a detection system monitors electrical quantities on synchrophasors, models their behavior, and estimates typical energy consumption. Other approaches observe anomalies on controllers [26], [27], or model probability distribution of transmitted packets (inter-arrival time, size, occurrence of specific packets) [28], [29], [30].

Our research deals with network monitoring of ICS packets transmitted over smart grid links rather than observing electrical quantities on the process bus or monitoring end devices. As demonstrated in our previous work [31], real-time visibility of ICS data exchange helps revealing unexpected ICS commands, unusual command frequency, or suspicious command order that would stay unnoticed by common detection techniques. Thus, we limit the overview of related works to those approaches that work on network layer and observe ICS packets.

In their work, Lin and Nadjm-Tehrani [32], [33] model inter-arrival times of IEC 104 spontaneous events using a probabilistic suffix tree and categorize the traffic into five different groups based on the periodicity and stability. Using probabilistic suffix trees they predict the future behavior of the IEC 104 communication and detect possible deviations. The method is computationally intensive and sensitive to network delays. In our work we do not consider time but model semantics of ICS commands by DPAs, which makes the system more robust.

The representation of Modbus communication using finite state machines was introduced by Goldenberg and Wool [34]. The authors extract key fields from Modbus packets and encode them as alphabet symbols. Automata transitions express the behavior of the system in terms of the sequence of expected message exchanges. Like our approach, their model can recognize invalid and unexpected messages. Our model is able to accommodate more aspects of communication, especially the expected frequency of exchanged messages.

Caselli et al. in [35], [36] employ discrete-time Markov chains to represent various industrial communication. The messages with the same semantics, e.g., read coils from address 0 for Modbus, are grouped to states. Transitions

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

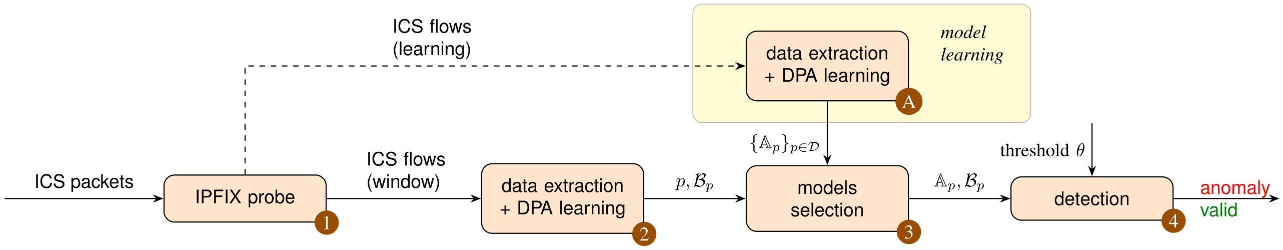


Fig. 2. A high-level overview of the proposed detection approach, with offline model learning part the highlighted.

represent a sequence of messages, which is in principle similar to our model. Instead of clustering, we learn the probabilistic automaton from all message sequences representing a normal behavior. Our approach is fully automated, not requiring to specify the semantics of each command.

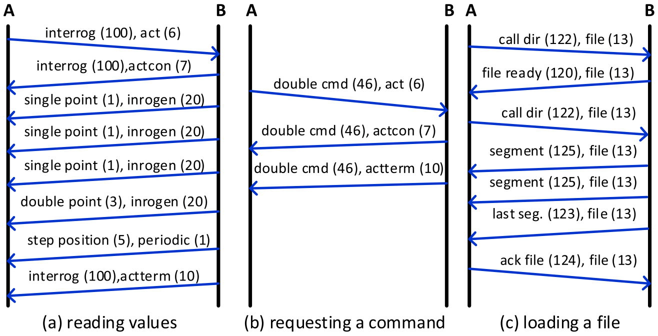
The idea of deriving a communication model from network traces was also discussed by Wang et al. [37] even if it was not used for industrial communication. The authors create a probabilistic protocol state machine to represent application protocols such as SMTP. During the learning process, messages are extracted from network traces and grouped according to the calculated similarity. The idea of their automata model is close to the deterministic probability automata introduced by De La Higuera [20]. However, their goal is different—they try to extract specifications for possibly unknown protocols.

To detect attacks, Krueger et al. [38] developed a method for protocol inspection and state machine analysis that infers a state machine and protocol message format from network traffic. They create a Hidden Markov Model from protocol messages represented as *n-grams* and their probability of occurrence in communication. The model is employed to simulate the behavior of honeypots. Unlike us, their Markov models are large (hundred of states) even when minimization is applied. In addition, our model is easier to compute.

III. Extended Automata-Based Anomaly Detection

In this section, we present our method of detecting anomalies in ICS communication. Sections III-A to III-D recall the basics of the method published in our previous papers [31], [39]. Sections III-E to III-G describe our novel contributions. The section is structured as follows:

- Section III-A recalls how input data are obtained from an IPFIX monitoring probe that collects IPFIX flows and partitioned them into conversations, i.e., sequences of messages exchanged between pairs of devices within a single communication session.
- Section III-B explains how to interpret input conversations as multisets of words over a finite alphabet and how to model these multisets using a probabilistic model in the form of DPAs.
- Section III-C overviews the general architecture of the diagnostic system, including the DPA learning from a valid traffic samples and the detection which is thoroughly described in two subsequent sections.


A
B
interrog (100), act (6)
interrog (100),actcon (7)
single point (1), inrogen (20)
single point (1), inrogen (20)
single point (1), inrogen (20)
double point (3), inrogen (20)
step position (5), periodic (1)
interrog (100),actterm (10)
A
B
double cmd (46), act (6)
double cmd (46), actcon (7)
double cmd (46), actterm (10)
A
B
call dir (122), file (13)
file ready (120), file (13)
call dir (122), file (13)
segment (125), file (13)
segment (125), file (13)
last seg. (123), file (13)
ack file (124), file (13)

(a) reading values (b) requesting a command (c) loading a file

Fig. 3. Examples of IEC 104 conversations.

- Section III-D recalls the technical basis of the detection with a single DPA model representing the valid traffic, as used in [31], [39].
- Section III-E describes our novel *multiple-model detection* that improves precision and reduces false positives. Unlike a single DPA model, it is composed of multiple DPAs representing behavior of the normal traffic.
- Section III-F presents a novel technique of *model reduction* that keeps the run-time computational resources of the multiple-model reduction practically manageable.
- Section III-G finally describes a novel method of *generating diagnostic traces* that help to identify the cause of a reported anomaly.

A high-level overview of the presented method is graphically expressed in Fig. 2.

A. Application-Level Monitoring of ICS Communication

In [31], [39], we implemented an extension to IPFIX monitoring probe (❶ in Fig. 2) that collects IPFIX flows enriched with domain-specific attributes from ICS packets. In [14] we analyzed semantics of common smart grid protocols and identified a sufficient set of attributes for application-level monitoring: ASDU type (`AsduType`) and Cause of Transmission (`CoT`) for IEC 104, MMS Type and Service for MMS packets, Application ID, and Control Block Reference for GOOSE, and Type, Class ID and OBIS code for DLMS. Based on flows, we define ICS conversations as *logical sequences of ICS messages exchanged between pairs of devices*, see Fig. 3.

Example 1: Consider three IEC 104 conversations exchanged between stations A and B as depicted in Fig. 3. Conversation (a) reads a set of values, conversation (b) requests a command execution, and conversation (c) transmits a file. Notice that each IEC 104 message is represented

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

by a pair $\langle \texttt{AsduType}, \texttt{CoT} \rangle$. Thus, IEC 104 conversations between stations A and B form three command sequences $u = \langle 100, 6\rangle\langle 100, 7\rangle\langle 1, 20\rangle\langle 1, 20\rangle\langle 1, 20\rangle\langle 3, 20\rangle\langle 5, 1\rangle \langle 100, 10\rangle$, $v = \langle 46, 6\rangle\langle 46, 7\rangle\langle 46, 10\rangle$, $w = \langle 122, 13\rangle\langle 120, 13\rangle\langle 122, 13\rangle\langle 125, 13\rangle\langle 125, 13\rangle\langle 123, 13\rangle\langle 124, 13\rangle$.

B. Modeling ICS Conversations by Probabilistic Automata

We will now discuss the general method we use to learn a model of a normal ICS traffic. Namely, we learn DPA from a sample of ICS conversations that represent a normal system communication (see Ⓐ in Fig. 2).

We adopt the theory of DPA of [20]. The ICS messages play the role of *symbols* and form a finite alphabet Σ, with Σ^* denoting all finite strings over Σ (i.e., all possible ICS conversations). A *deterministic probabilistic automaton (DPA)* is then a tuple $\mathcal{A} = (Q, \delta, q_0, \mathbb{F})$ where Q is a finite set of *states*, $\delta : Q \times \Sigma \times Q \to [0, 1]$ is a (total) *transition function* assigning probabilities from the interval $[0, 1]$ of rational numbers to *transitions*, $q_0 \in Q$ is the *initial state*, and $\mathbb{F} : Q \to [0, 1]$ is a mapping that assigns the *acceptance probabilities* to states. The automaton must be *consistent*, meaning that for each state $q \in Q$, the sum of probabilities of the outgoing transitions plus the probability of acceptance is 1, formally, $\mathbb{F}(q) + \sum_{a \in \Sigma, r \in Q} \delta(q, a, r) = 1$. The automaton is implicitly *deterministic*, hence every state $q \in Q$ has a unique successor via every symbol $a \in \Sigma$, i.e., $|\{r | \delta(q, a, r) > 0\}| = 1$. The automaton defines a *probability distribution* $\mathcal{P}_{\mathcal{A}} : \Sigma^* \to [0, 1]$ over Σ^* as follows. Each string $w = a_1 \ldots a_n \in \Sigma^*$ has its unique *trace*, the sequence $\pi = (q_0, a_1, q_1) \cdots (q_{n-1}, a_n, q_n)$ where $\delta(q_{i-1}, a_i, q_i) > 0$ for $1 \leq i \leq n$, and its probability is defined based on the trace as $\mathcal{P}_{\mathcal{A}}(w) = \mathbb{F}(q_n) \cdot \prod_{1 \leq i \leq n} \delta(q_{i-1}, a_i, q_i)$. Informally, $\mathcal{P}_{\mathcal{A}}(w)$ is a probability of the random walk through the automaton that respects symbols of w and is accepted by the end state.

A sample of typical communication between ICS devices, in the form of a multiset of ICS conversations, is given to the algorithm Alergia [20] extended with an automated estimation of parameters (see [15] for details). Alergia then learns a DPA that represents a generalisation of the probabilistic distribution of conversations in the sample. This is a general method that can be used to learn the behavior of any system that uses any smart grid communication protocol.

Example 2: Fig. 4 shows a DPA $\mathcal{A}$ learned from IEC 104 conversations that include a file transfer ($\texttt{AsduType} \in \{120, \ldots, 125\}$, $\texttt{CoT} = 13$) and spontaneous events ($\texttt{AsduType} = 36, \texttt{CoT} = 3$). The automaton $\mathcal{A}$ represents the language by which two IEC 104 devices talk. The conversation $w = \langle 122, 13\rangle \ \langle 120, 13\rangle \ \langle 122, 13\rangle \ \langle 125, 13\rangle \ \langle 123, 13\rangle \ \langle 124, 13\rangle$ has the probability $\mathcal{P}_{\mathcal{A}}(w) = 0.54 \cdot 1 \cdot 1 \cdot 0.5 \cdot 0.06 \cdot 1 \cdot 0.5 = 0.0081$. Its proper prefixes have probability 0 since they reach states with 0 acceptance probability. Any IEC 104 conversation w that is not a part of the specified device-to-device language would not be accepted by $\mathcal{A}$ and receives probability $\mathcal{P}_{\mathcal{A}}(w) = 0$.

C. Architecture of the Diagnostic System

The schema of the system is shown in Fig. 2. Let $\mathcal{D}$ denote the set of all pairs of devices that communicate within the network. DPAs representing valid traffic for each communication pair $p \in \mathcal{D}$ in the system are created during the learning phase (Ⓐ). In the detection phase, the incoming traffic is split into time windows with a fixed-length but configurable duration (default is 5 min). The traffic in form of flow records is provided by the IPFIX probe (①). Conversations for each communication pair p (identified by IP addresses and ports) are selected, and DPA $\mathcal{B}_p$ representing the conversations for p is computed (②). In the following step, models $\mathbb{A}_p$ representing valid traffic of a communication pair p are selected (③). In the last step, the detection mechanism is applied on $\mathbb{A}_p$ and $\mathcal{B}_p$. By evaluating probability of tested conversations it either approves their validity or returns a diagnostic trace of an anomaly (④).

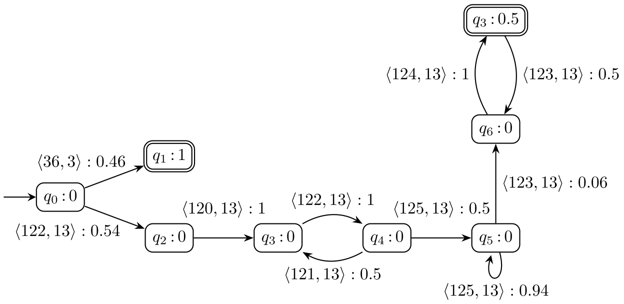

$q_3 : 0.5$
$\langle 124, 13\rangle : 1$
$\langle 123, 13\rangle : 0.5$
$q_6 : 0$
$\langle 36, 3\rangle : 0.46$
$q_1 : 1$
$\langle 123, 13\rangle : 0.06$
$q_0 : 0$
$\langle 120, 13\rangle : 1$
$\langle 122, 13\rangle : 1$
$\langle 125, 13\rangle : 0.5$
$\langle 122, 13\rangle : 0.54$
$q_2 : 0$
$q_3 : 0$
$q_4 : 0$
$q_5 : 0$
$\langle 121, 13\rangle : 0.5$
$\langle 125, 13\rangle : 0.94$

Fig. 4. DPA $\mathcal{A}$ representing IEC 104 communication. Here and in all the other figures of DPAs, we depict the accepting probability inside each state and highlight states with nonzero acceptance probability. We label transitions in the form *symbol:probability*. In case of IEC 104 communication, *symbol* includes two IEC 104 packet header values, namely ASDU type and Cause of Transmission.

D. Single-Model Anomaly Detection

Our starting point is the detection procedure of [39] that receives as input a single DPA $\mathcal{A}_p$ representing valid traffic for a given communication pair p and a DPA $\mathcal{B}_p$ describing an incoming traffic captured within a fixed-size time *window under scrutiny*. The detection stands for measuring the difference between $\mathcal{B}_p$ and the single model $\mathcal{A}_p \in \mathbb{A}_p$. Since DPAs express probability (frequency) of strings, it is natural to compare $\mathcal{P}_{\mathcal{A}_p}(w)$ and $\mathcal{P}_{\mathcal{B}_p}(w)$ for each $w \in \Sigma^*$. We use the 2-Euclid distance (or just Euclid distance) defined as

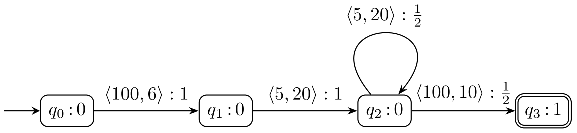
$$L_2(\mathcal{A}_p, \mathcal{B}_p) = \sqrt{\sum_{w \in \Sigma^*} \left(\mathcal{P}_{\mathcal{A}_p}(w) - \mathcal{P}_{\mathcal{B}_p}(w)\right)^2} \quad (1)$$

Intuitively, the Euclid distance sums the differences of probabilities assigned to strings by $\mathcal{A}_p$ and $\mathcal{B}_p$. A threshold parameter θ controls how different the two automata must be to get an anomaly, i.e., $L_2(\mathcal{A}_p, \mathcal{B}_p) > \theta$. Low value of θ causes higher possibility of false alarms (false positives), high value of θ can let some anomalies be undetected (false negatives). We experimentally found (see Section IV) suitable values of θ in range 0.1 and 0.25.[2]

Example 3: Consider the DPA $\mathcal{A}_p$ representing a desired IEC 104 behaviour given in Fig. 5. The automaton expresses

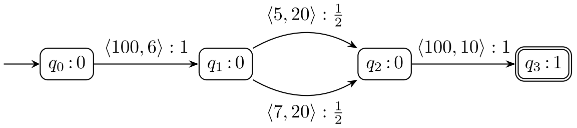
[2] This range is a trade-off between the accuracy and the false positivity rate. Since we assume stable communication, the value 0.1 provides enough space to cover possible short-term disturbances in the traffic. On the other hand, threshold above 0.25 could possibly miss some crucial anomalies.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.



$q_0:0$
$\langle 100,6\rangle : 1$
$q_1:0$
$\langle 5,20\rangle : 1$
$q_2:0$
$\langle 5,20\rangle : \frac{1}{2}$
$\langle 100,10\rangle : \frac{1}{2}$
$q_3:1$

Fig. 5. A DPA $\mathcal{A}_p$ representing a valid IEC 104 traffic (simplified for the purpose of Example 3, see Fig. 4 for a realistic example of a traffic model).



$q_0:0$
$\langle 100,6\rangle : 1$
$q_1:0$
$\langle 5,20\rangle : \frac{1}{2}$
$\langle 7,20\rangle : \frac{1}{2}$
$q_2:0$
$\langle 100,10\rangle : 1$
$q_3:1$

Fig. 6. A DPA $\mathcal{B}_p$ representing a time window.

the usual traffic that includes conversations differing on the number of messages $\langle 5, 20\rangle$, i.e., the valid traffic contains conversations $\{u_n | u_n = \langle 100, 6\rangle\langle 5, 20\rangle^n\langle 100, 10\rangle, n>1\}$. Longer conversations have smaller probability.

Further consider the DPA $\mathcal{B}_p$ given in Fig. 6 created from a time window with the testing IEC 104 traffic that includes a set of conversations $w_1 = u_1 = \langle 100, 6\rangle\langle 5, 20\rangle\langle 100, 10\rangle$ and $w_2 = \langle 100, 6\rangle\langle 7, 20\rangle\langle 100, 10\rangle$ that occur with the same probability 50%.

The L_2 distance between these two DPAs is given as

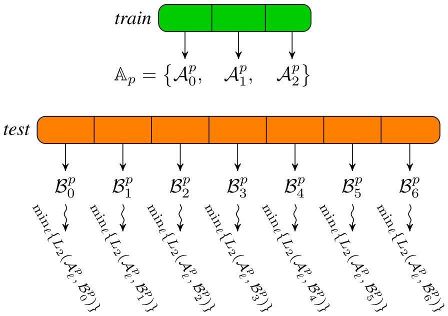
$$\begin{aligned}L_2^2(\mathcal{A}_p, \mathcal{B}_p) &= \left(\mathcal{P}_{\mathcal{A}_p}(u_1) - \mathcal{P}_{\mathcal{B}_p}(u_1)\right)^2\\ &\quad + \left(\mathcal{P}_{\mathcal{A}_p}(w_2) - \mathcal{P}_{\mathcal{B}_p}(w_2)\right)^2\\ &\quad + \left(\mathcal{P}_{\mathcal{A}_p}(u_2) - \mathcal{P}_{\mathcal{B}_p}(u_2)\right)^2\\ &\quad + \left(\mathcal{P}_{\mathcal{A}_p}(u_3) - \mathcal{P}_{\mathcal{B}_p}(u_3)\right)^2 + \cdots\\ &= 0^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{8}\right)^2 + \cdots\end{aligned}$$

Hence, $L_2(\mathcal{A}_p, \mathcal{B}_p) = \sqrt{\frac{1}{4} + \frac{1}{12}} \approx 0.57$. Intuitively, $\mathcal{A}_p$ and $\mathcal{B}_p$ describe a different behaviour, because $\mathcal{B}_p$ represents a traffic where half of the conversations are w_2, which has zero probability in the model $\mathcal{A}_p$. The L_2 distance confirms the intuition, since 0.57 indicates an anomaly.

By definition, the sum of Euclid distances is over all strings, however, the distance L_2 can be computed in a polynomial time. The algorithm uses a matrix representation of probabilistic automata and expresses the infinite sum in a closed form, see [40].

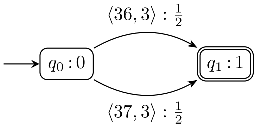
E. Multiple-Model Anomaly Detection

The single-model detection suffers from an excessive number of false positives (hundreds or thousands daily in our experiments). Their source is a long term variability of ICS communication—its characteristics may depend on factors such as the time of the day, mode of use of the system, etc. Since the scrutinised communication windows are short-term, they vary depending on the current conditions. The single-model approach however summarises all the variability into a single DPA $\mathcal{A}_p$. Even a perfectly normal short time window may hence differ from the learned DPA significantly, i.e., their distance $L_2(\mathcal{A}_p, \mathcal{B}_p)$ may be large, in which case the detection generates a false positive.



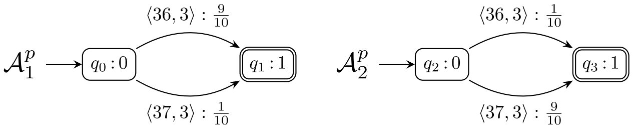
train
$\mathbb{A}_p = \{\mathcal{A}_0^p, \mathcal{A}_1^p, \mathcal{A}_2^p\}$
test
$\mathcal{B}_0^p$
$\mathcal{B}_1^p$
$\mathcal{B}_2^p$
$\mathcal{B}_3^p$
$\mathcal{B}_4^p$
$\mathcal{B}_5^p$
$\mathcal{B}_6^p$
$\min_\ell\{L_2(\mathcal{A}_\ell^p, \mathcal{B}_0^p)\}$
$\min_\ell\{L_2(\mathcal{A}_\ell^p, \mathcal{B}_1^p)\}$
$\min_\ell\{L_2(\mathcal{A}_\ell^p, \mathcal{B}_2^p)\}$
$\min_\ell\{L_2(\mathcal{A}_\ell^p, \mathcal{B}_3^p)\}$
$\min_\ell\{L_2(\mathcal{A}_\ell^p, \mathcal{B}_4^p)\}$
$\min_\ell\{L_2(\mathcal{A}_\ell^p, \mathcal{B}_5^p)\}$
$\min_\ell\{L_2(\mathcal{A}_\ell^p, \mathcal{B}_6^p)\}$

Fig. 7. Visualization of the proposed anomaly detection.



$q_0:0$
$\langle 36,3\rangle : \frac{1}{2}$
$\langle 37,3\rangle : \frac{1}{2}$
$q_1:1$

Fig. 8. A single DPA $\mathcal{A}_p$ representing the valid IEC 104 traffic.



$\mathcal{A}_1^p$
$q_0:0$
$\langle 36,3\rangle : \frac{9}{10}$
$\langle 37,3\rangle : \frac{1}{10}$
$q_1:1$
$\mathcal{A}_2^p$
$q_2:0$
$\langle 36,3\rangle : \frac{1}{10}$
$\langle 37,3\rangle : \frac{9}{10}$
$q_3:1$

Fig. 9. A set of DPAs $\{\mathcal{A}_1^p, \mathcal{A}_2^p\}$ representing the valid traffic.

To overcome this drawback, we improve our method by learning *multiple models* (Ⓐ) that consist of a set of DPAs $\mathbb{A}_p = \{\mathcal{A}_i^p\}_{i\in\mathcal{I}}$. In our experiments, we divide the training traffic into fixed-length-duration subparts (where the duration corresponds to the size and double size of the time window used during the detection). Each such created part is an input of learning yielding a DPA from $\mathbb{A}_p$. A detection algorithm (④) first finds the *closest model* $\mathcal{A}_\ell^p \in \mathbb{A}_p$ to the window under scrutiny according to its Euclidian distance. Anomaly alert is raised if the distance of the closest model $\mathcal{A}_\ell^p$ from $\mathcal{B}_p$ is greater than the defined *anomaly threshold* θ, formally:

$$\min_{\mathcal{A}_p\in\mathbb{A}_p} L_2(\mathcal{A}_p, \mathcal{B}_p) > \theta. \tag{2}$$

For a better illustration, the proposed detection method is visualized in Fig. 7 and the effect of multiple models is demonstrated in the following example.

Example 4: Assume that the first part of a simple valid IEC 104 traffic related to a single device consists of spontaneous conversations $\langle 36, 3\rangle$ and $\langle 37, 3\rangle$ with a dominance of the first mentioned conversations. In the second part, the ratio changes in favor to the conversations of the second type. If we represent the traffic using a single model only, we get the automaton $\mathcal{A}_p$ given in Fig. 8.

On the other hand, if we apply learning on each part separately, we obtain the DPAs $\mathcal{A}_1^p$ and $\mathcal{A}_2^p$ given in Fig. 9 assigning different probabilities for each conversation. As you can see, the second representation captures subtle difference in the learning traffic, in this case the change of occurrence of different spontaneous events.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

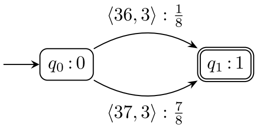

$\langle 36, 3\rangle : \frac{1}{8}$
$q_0 : 0$
$q_1 : 1$
$\langle 37, 3\rangle : \frac{7}{8}$

Fig. 10. DPA $\mathcal{B}_p$ representing the incoming window.

Further assume that the incoming traffic window contains one conversation $\langle 36, 3\rangle$ and seven conversations $\langle 37, 3\rangle$. A DPA $\mathcal{B}_p$ learned for this traffic is shown in Fig. 10.

If we apply the single-model anomaly detection according to $\mathbb{A} = \{\mathcal{A}_p\}$, we obtain the value $L_2(\mathcal{A}_p, \mathcal{B}_p) \approx 0.53$. If we apply the multiple-model detection with $\mathbb{A} = \{\mathcal{A}^p_1, \mathcal{A}^p_2\}$, we get the value $\min\{L_2(\mathcal{A}^p_1, \mathcal{B}_p), L_2(\mathcal{A}^p_2, \mathcal{B}_p)\} \approx 0.035$. Hence, by applying the single-model detection according to the whole traffic (represented by $\mathcal{A}_p$) with the threshold $\theta = 0.2$, we get an anomaly alert, which is not desirable in this case.

F. Model Reduction

According to Eq. (2), the inspection of a each scrutinised window requires evaluating the distance from every automaton in $\mathbb{A}_p$. Since the number of automata in $\mathbb{A}_p$ easily reaches hundreds, this task becomes impractically expensive. We hence need a method of reducing the number of models, preferably while preserving precision of the detection.

To this end, we utilise the fact that conversations between two ICS devices are mostly the same or very similar. This means that also automata representing such ICS traffic are equal or at least similar, e.g., they only have slightly different probabilities of certain strings. Thus, we can reduce redundant computation by removing similar automata from a set of models $\mathbb{A}_p$. To decide which automaton is to be removed we define the error function that measures how good is the resulting set of DPAs. The function that computes the detection error introduced by reducing the model from $\mathbb{A}$ to $\mathbb{A}'$ is defined as follows:

$$error(\mathbb{A}, \mathbb{A}') = \max_{\mathcal{A}_1 \in \mathbb{A} \setminus \mathbb{A}'} \left\{ \min_{\mathcal{A}_2 \in \mathbb{A}'} L_2(\mathcal{A}_1, \mathcal{A}_2) \right\}. \tag{3}$$

The goal is to find a subset $\mathbb{A}' \subseteq \mathbb{A}$ such that the error of $error(\mathbb{A}, \mathbb{A}')$ is below a given threshold ϵ.[3] The error function has the property that if we use $\mathbb{A}'_p \subseteq \mathbb{A}_p$ as a model for the anomaly detection, the value, based on the procedure decides whether the window contains anomaly or not, differs by $error(\mathbb{A}_p, \mathbb{A}'_p)$ in the worst case. In particular, assume that $\mathcal{A}_p \in \mathbb{A}$ is the closest model to an automaton $\mathcal{B}_p$ representing the input traffic window. Moreover, based on the error function, there is some $\mathcal{A}'_p \in \mathbb{A}'$ s.t. $L_2(\mathcal{A}_p, \mathcal{A}'_p) \leq \epsilon$. From the triangle inequality (L_2 is a distance), we get $L_2(\mathcal{A}'_p, \mathcal{B}_p) \leq L_2(\mathcal{A}_p, \mathcal{B}_p) + L_2(\mathcal{A}_p, \mathcal{A}'_p) \leq L_2(\mathcal{A}_p, \mathcal{B}_p) + \epsilon$, which defines the upper bound.

The reduction procedure is described by Algorithm 1. Since the optimal computation of $\mathbb{A}'$ is computationally infeasible, we use a greedy algorithm iteratively saturating a set of automata that can be removed. In each iteration the algorithm selects an automaton $\mathcal{A}$ having the smallest distance from another automaton that has not been removed yet. If the error caused by removing this automaton is below ϵ the removal is approved. Note that for $\epsilon = 0$ we remove from $\mathbb{A}_p$ automata that are equivalent. The model reduction is a part of the offline learning (Ⓐ) without any negative impact on real-time anomaly detection.

Algorithm 1: Model Reduction

```
Input:
  $\mathbb{A}$ – set of DPAs representing the original model
  $\epsilon$ – threshold value for reduction
Result: $\mathbb{A}' \subseteq \mathbb{A}$ s.t. $error(\mathbb{A}, \mathbb{A}') \leq \epsilon$
$R := R' := \emptyset$ ;                          ▷ sets of DPAs to be removed
while $error(\mathbb{A}, \mathbb{A} \setminus R') \leq \epsilon$ do
    $R := R'$;
    Select $\mathcal{A} \in \mathbb{A} \setminus R$ s.t. $\min_{\mathcal{B} \in \mathbb{A} \setminus R, \mathcal{B} \neq \mathcal{A}} L_2(\mathcal{A}, \mathcal{B})$ is minimal;
    $R' := R \cup \{\mathcal{A}\}$ ;                ▷ update $R'$ with the DPA $\mathcal{A}$
return $\mathbb{A}' := \mathbb{A} \setminus R$;
```

G. Generating Diagnostic Traces

The anomaly report of the original method of [15] includes only the scrutinised window and its L_2 distance from the learned DPA. It would be a task of the operator to inspect the window and decide whether or not it is indeed anomalous and why. This would require a high level of expertise as well as time. Therefore, we developed a method of generating succinct diagnostic traces that capture a reason of anomaly.

Assume a particular window consisting of a multiset of conversations $\mathcal{C}_p$ and triggering the anomaly alert. The reasoning is then performed on the level of conversations $\mathcal{C}_p$, DPA $\mathcal{B}_p$ obtained from $\mathcal{C}_p$, and the most accurate learned model $\mathcal{A}_p$, which is given using the following formula (cf. Section III-E):

$$\mathcal{A}_p \in \mathbb{A}_p \text{ s.t. } L_2\big(\mathcal{A}_p, \mathcal{B}_p\big) \text{ is minimal.} \tag{4}$$

The first output of the detection method is a set of conversations $\mathcal{E}_{bad}$ occurring within the suspicious window that do *not* occur in the model:

$$\mathcal{E}_{bad} = \big\{ w \in \mathcal{C}_p | \mathcal{P}_{\mathcal{A}_p}(w) = 0 \big\}. \tag{5}$$

Since our approach is even able to detect anomalies based on missing communication, we provide also an example of a conversation that was not present in the window but it should be because $\mathcal{A}_p$ assigns to the conversation nonzero probability. As the number of such conversations may be infinite, we pick conversations with the highest probability only. In particular, we first compute the set

$$\mathcal{L} = \big\{ w \in \Sigma^* | \mathcal{P}_{\mathcal{A}_p}(w) > 0, \mathcal{P}_{\mathcal{B}_p}(w) = 0 \big\}. \tag{6}$$

The set is regular, and hence it can be effectively represented by a finite automaton. Based on this set, we find the most probable strings from $\mathcal{L}$ in $\mathcal{A}_p$, which is an example of a missing conversation. Formally,

$$\mathcal{E}_{miss} = \left\{ w \in \mathcal{L} | \mathcal{P}_{\mathcal{A}_p}(w) = \max_{u \in \mathcal{L}} \mathcal{P}_{\mathcal{A}_p}(u) \right\}. \tag{7}$$

[3] Note that similarity threshold ϵ is different from detection threshold θ.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

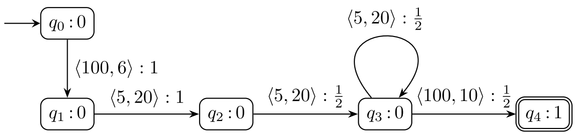

$q_0 : 0$
$\langle 100, 6\rangle : 1$
$q_1 : 0$
$\langle 5, 20\rangle : 1$
$q_2 : 0$
$\langle 5, 20\rangle : \frac{1}{2}$
$q_3 : 0$
$\langle 5, 20\rangle : \frac{1}{2}$
$\langle 100, 10\rangle : \frac{1}{2}$
$q_4 : 1$

Fig. 11. Product automaton $\mathcal{G}$.

Roughly speaking, to compute $\mathcal{E}_{miss}$ we first need to represent the language $\mathcal{L}$ using a deterministic finite automaton $\mathcal{A}_{\mathcal{L}}$. Note that the number of states of $\mathcal{A}_{\mathcal{L}}$ is bounded by $|\mathcal{A}_p| \cdot |\mathcal{B}_p|$. In the second step, we restrict $\mathcal{A}_p$ to words from $\mathcal{L}$ only. This can be done using a product of $\mathcal{A}_p$ and $\mathcal{A}_{\mathcal{L}}$ with keeping the probabilities from $\mathcal{A}_p$. In this case, the product need not to be a DPA, because the consistency condition might be violated. We denote the product as $\mathcal{G} = (Q, \delta, q_0, \mathbb{F})$. In order to obtain $\mathcal{E}_{miss}$ we iteratively compute the least fixpoint of the system representing a most probable path in the automaton:

$$\ell_q = \mathbb{F}(q), \tag{8}$$

$$\ell_q = \max_{p\in Q, a\in\Sigma}\left\{\ell_q, \delta(q, a, p) \cdot \ell_p\right\} \tag{9}$$

Together with computation of the most probable path, we also store information about the concrete most probable strings labelling the paths. In particular, at the beginning we set $W_q = \emptyset$ for $\mathbb{F}(q) = 0$ and $W_q = \{\epsilon\}$ otherwise. Then, every time ℓ_q is improved by a transition (q, a, p) and moreover $\ell_q = \delta(q, a, p) \cdot \ell_p$ holds before the assignment, we set $W_q = W_q \cup a.W_p$. If $\ell_q < \delta(q, a, p) \cdot \ell_p$ holds before the assignment, we set $W_q = a.W_p$. We have $\mathcal{E}_{miss} = W_{q_0}$ in the fixpoint.

Example 5: Consider two DPAs $\mathcal{A}_p$ and $\mathcal{B}_p$ from Example 3. Then, the language $\mathcal{L}$ can be described by the regular expression $\langle 100, 6\rangle\langle 5, 20\rangle\langle 5, 20\rangle^{+}\langle 100, 10\rangle$. The product $\mathcal{G}$ restricting $\mathcal{A}_p$ to $\mathcal{L}$ is then given in Fig. 11. After the fixpoint computation, we obtain $\ell_{q_0} = \frac{1}{4}$ and $W_{q_0} = \{\langle 100, 6\rangle\langle 5, 20\rangle \langle 5, 20\rangle\langle 100, 10\rangle\}$ where q_0 is the initial state of $\mathcal{G}$.

IV. Experimental Evaluation

This section aims at the experimental evaluation of the method using various available datasets for (i) evaluation of the accuracy and efficiency of the detection and (ii) measuring the effect of the proposed extensions in terms of false positives. First, we introduce our datasets and evaluation method. Then we show experimental results of automata reduction. The we present our experiments with IEC 104 and MMS communication, respectively. Finally, we discuss the potential of our method for real-world deployment.

A. Used Tools and Benchmarks

The presented anomaly detection approach was implemented as the Python tool Detano,[4] which is an extended implementation of our previous tool, here denoted as Autanom [15]. Our experiments were conducted on benchmarks containing datasets provided by various sources: (i) IEC 104 traffic generated by Brno University of Technology and containing various attack scenarios [15] (this benchmark is denoted as BUT and is available at IEEE Dataport [41]), (ii) IEC 104 traffic provided by RTSLab, Linköping University, Sweden [42] (denoted as RTS), (iii) IEC 104 traffic generated in a virtual ICS network (denoted as VRT and also available at [41]), (iv) MMS traffic provided by European Network for Cyber Security[5] (denoted as ENCS), and (v) MMS traffic provided by G-ICS labs, Université Grenoble Alpes France [43] that was further enhanced by various attack scenarios, as described below (denoted as GICS).

TABLE I
Overview of the VRT Benchmark [41]

Dataset	Duration	Packets	MITRE Technique ID
HMI-to-IEC104	1 day 21 hours	195,424	normal traffic
IEC104	1 day 21 hours	135,298	normal traffic
Scada-to-Sub	1 day 21 hours	60,126	normal traffic
RTU-MITM	3 days 3 hours	258,582	T0830,T0831
HMI-MITM	3 days 3 hours	557,215	T0830,T0831
HMI-report-block	1 day 37 min	184,334	T0803,T0804
HMI-replay	1 day 5 hours	248,394	T0831,T0856
HMI-value-change	1 day 10 hours	261,967	T0831,T0832,T0836
HMI-masquerading	22 hours	164,760	T0831,T0856

TABLE II
Overview of the ENCS and GICS Benchmarks

Dataset	Duration	Packets	MITRE Technique ID
inside-substation	15 min	1,558	T0802,T0831,T0840
gics-lost-connection	1 hour 35 min	2,581	outage
gics-modified	1 hour 35 min	2,706	T0831,T0855,T0856
gics-scanning	1 hour 35 min	2,778	T0801,T0802,T0856
gics-interrupt	1 hour 35 min	2,722	T0855,T0858,T0881

Attacks occurring within the VRT benchmark, in particular HMI-* datasets, were created at our University by penetration tools that inserted spoofed or modified IEC 104 messages on the communication link. Attacks on MMS communication within the GICS benchmark were created manually by inserting spoofed MMS packets into generated traces. These attack traces were inspired by published attack scenarios. For our experiments we used traces obtained by the Flowmon IPFIX probe[6] from the captured PCAP files.

Overview of the datasets occurring within the VRT benchmark is shown in Table I.

These datasets contain up to 3 days of traffic with hundred thousands ICS packets.

Datasets included in the ENCS and GICS benchmarks are listed in Table II.

These datasets contain up to 1.5 hours of the traffic with thousands of ICS packets.

The datasets from the BUT benchmark contain up to 3 days of traffic with more than one million of ICS packets.[7] The datasets occurring within the RTS benchmark contain up to 12 days of traffic with more than 3 million packets, see Table III. More detailed description of of these datasets is in [15].

[4] Available at https://github.com/vhavlena/detano [Feb 2022].

[5] See https://encs.eu/ [Feb 2022].

[6] See https://www.flowmon.com/en/products/appliances/probe.

[7] BUT traces are available at https://github.com/matousp/datasets [Jun 2022].

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

TABLE III
OVERVIEW OF THE BUT AND RTS BENCHMARKS

Dataset	Duration	Packets	Technique ID
`10122018-104Mega`	4 hours 53 min	76,865	normal traffic
`13122018-mega104`	2 days 23 hours	1.073,732	normal traffic
`mega104-14-12-18`	15 hours 38 min	14,429	normal traffic
`mega104-17-12-18`	2 days 19 hours	58,929	normal traffic
`injection-attack`	2 days 19 hours	58,930	T0805-6,T0881
`rogue-device`	2 days 19 hours	58,893	T0848
`dos-attack`	2 days 19 hours	58,932	T0814
`scanning-attack`	2 days 19 hours	58,927	T0801-2,T0840
`switching-attack`	2 days 19 hours	59,002	T0838,T0858
`connection-loss`	2 days 19 hours	57,863	–
`10days2`	12 days 21 hours	882,854	normal traffic
`repaired-rtu8novlan`	6 days 18 hours	3.117,663	normal traffic
`repaired-rtu11novlan`	6 hour 18 min	1.828,733	normal traffic

TABLE IV
MITRE TACTICS OBSERVABLE BY ICS FLOW MONITORING. THE TECHNIQUES WITH (*) ARE A SUBJECT OF OUR EXPERIMENTS

Tactics	Technique	Technique ID
Initial Access	Internet Accessible Device	T0883
Initial Access	Rogue Master	T0848*
Execution	Change Operating Mode	T0858*
Evasion	Spoof Reporting Message	T0856*
Discovery	Network Connection Enum.	T0840*
Discovery	Remote System Discovery	T0846*
Discovery	Remote System Info Discovery	T0888*
Collection	Automated Collection	T0802*
Collection	Detect Operating Mode	T0868*
Collection	Man in the Middle	T0830*
Collection	Monitor Process State	T0801*
Collection	Point & Tag Identification	T0861
Collection	Program Upload	T0845
Command and Control	Commonly Used Port	T0885
Command and Control	Connection Proxy	T0884
Command and Control	Standard App Layer Protocol	T0869
Inhibit Resp. Function	Activate Firmware Update Mode	T0800
Inhibit Resp. Function	Alarm Suppression	T0878*
Inhibit Resp. Function	Block Command Message	T0803*
Inhibit Resp. Function	Block Reporting Message	T0804*
Inhibit Resp. Function	Denial of Service	T0814*
Inhibit Resp. Function	Modify Alarm Settings	T0838*
Inhibit Resp. Function	Service Stop	T0881*
Impair Process Control	Modify Parameter	T0836*
Impair Process Control	Spoof Reporting Message	T0856*
Impair Process Control	Unauthorized Command Message	T0855*
Impact	Manipulation of Control	T0831*
Impact	Manipulation of View	T0832*

The present anomalies are classified wrt. MITRE Adversarial Tactics, Techniques, and Common Knowledge framework for ICS (ATT&CK for ICS) [21], which deals with levels 1 and 2 of the Purdue model (PLC, RTU, HMI, etc.) [44]. The complete MITRE Matrix for ICS lists 12 tactics and 78 techniques (May 2022).

We identified 28 attacker's techniques according to MITRE ATT&CK for ICS that are observable in network communication and thus can be detected by the proposed automata-based method, see Table IV. Most of these techniques were implemented in our datasets (marked by $*$). This list clearly shows the scope of application of the proposed automata-based detection which depends on the visibility of an attack in the network communication. The selected techniques may occur in various stages of the Advanced Persistent Threats (APT) Kill Chain [45] which include Reconnaissance (Discovery), Weaponization (Initial Access), Exploitation (Collection), Install/Modify (Evasion, Inhibit Response Function), Command & Control, and Execute (Execution, Impair Process Control, Impact), see the first column of the table.

Obviously, the automata-based anomaly detection capability covers the attacker's activities related to sending unexpected commands, blocking ICS messages, scanning network resources, manipulating packets, etc. Since our model represents ICS commands, it can detect attacker's activities that cannot be recognized by statistical-based methods.

TABLE V
NUMBER OF MODELS USED FOR VARIOUS DATASETS

Dataset	Original	Model red. ($\epsilon = 0.0$)	Model red. ($\epsilon = 0.1$)
`HMI-MITM`	120	23 (19.1 %)	8 (6.6 %)
`RTU-MITM`	133	16 (12.03 %)	7 (5.2 %)
`HMI-to-IEC104`	163	13 (7.9 %)	4 (2.4 %)
`HM-replay`	917	27 (2.9 %)	14 (1.5 %)

B. Focus of the Experiments

In the experiments, we are particularly interested in the number of detected anomalies/attacks (true positives, denoted by "**Hit**"), the number of undetected anomalies (false negatives, denoted by "**Miss**"), the number of detected anomalies/attacks that are not true anomalies (false positives, denoted by "**FP**"), and the time required for processing a single window for a single communication pair.

An anomaly is detected if at least one window is reported as an anomaly during the anomaly occurrence. The number of false positives is then given as the number of windows wrongly marked as an anomaly. On the contrary, an anomaly is undetected (false negative) if no window is marked as a suspicious during the anomaly occurrence.

In the evaluation, we also measure the total number of DPAs representing all communication pairs $p \in \mathcal{D}$ in the smart grid network, given as $|\mathbb{A}| = \sum_{p\in\mathcal{D}} |\mathbb{A}_p|$.

C. Effect of Model Reduction

Before we move to the main part of experimental evaluation targeting efficiency and accuracy, we conduct experiments showing the effect of the proposed model reduction.

Table V shows the number of DPAs obtained during the DPA learning (Ⓐ) for a particular dataset. The column "**Original**" denotes the total number of DPAs learned from a particular dataset for each communication pair of devices. The column "**Model red.** ($\epsilon = 0.0$)" denotes the number of automata after removing the identical ones. Finally, the column "**Model red.** ($\epsilon = 0.1$)" denotes the number of automata left after the model reduction with the error bound $\epsilon = 0.1$. Removing identical automata has the biggest impact on the number of automata. However, the model reduction with $\epsilon = 0.1$ further notably decreases the number of automata, e.g., in the case of `HMI-to-IEC104` we could use for the detection only four DPAs instead of 13 (reduction

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

TABLE VI

SUMMARY RESULTS COMPARING DETANO AND AUTANOM ON IEC 104 BENCHMARKS. THE COLUMN "**WINDOWS**" DENOTES THE TOTAL NUMBER OF PROCESSED FIVE-MINUTES TIME WINDOWS ACROSS ALL COMMUNICATION PAIRS. THE COLUMN "**HIT**" DENOTES THE NUMBER OF DETECTED ATTACKS, "**MISS**" THE NUMBER OF ATTACKS THAT WERE NOT DETECTED, "**FP**" DENOTES THE NUMBER OF WINDOWS THAT WERE WRONGLY MARKED AS AN ATTACK, AND "$|\mathbb{A}|$" DENOTES THE TOTAL NUMBER OF MODEL DPAS FOR ALL COMMUNICATION PAIRS

Dataset	Windows	θ	Benchmark	DETANO				AUTANOM			
				Hit	Miss	FP	$\lvert\mathbb{A}\rvert$	Hit	Miss	FP	$\lvert\mathbb{A}\rvert$
10122018-104Mega	10	0.2	BUT	0	0	1	8	0	0	5	2
13122018-mega104	259	0.2	BUT	0	0	0	123	0	0	132	3
mega104-14-12-18	187	0.2	BUT	0	0	0	3	0	0	1	1
mega104-17-12-18	815	0.2	BUT	0	0	0	4	0	0	0	1
10days2	3,716	0.2	RTS	0	0	0	1	0	0	0	1
repaired-rtu8novlan	1,953	0.2	RTS	0	0	0	6	0	0	0	4
repaired-rtu11novlan	1,950	0.2	RTS	0	0	0	3	0	0	0	1
HMI-MITM	904	0.2	VRT	1	2*	0	23	3	0	75	1
HMI-to-IEC104	1,096	0.2	VRT	0	0	1	13	0	0	548	2
IEC104	548	0.2	VRT	0	0	0	9	0	0	0	1
RTU-MITM	904	0.2	VRT	1	2*	0	16	3	0	76	1
Scada-to-Sub	547	0.2	VRT	0	0	1	5	0	0	547	1
connection-loss	815	0.13	BUT	2	0	0	4	2	0	0	1
dos-attack	815	0.13	BUT	0	1	0	4	0	1	0	1
injection-attack	815	0.13	BUT	2	0	0	4	2	0	0	1
raw-device	827	0.13	BUT	1	0	0	4	1	0	0	1
scanning-attack	815	0.13	BUT	2	0	0	4	2	0	0	1
switching-attack	815	0.13	BUT	1	0	0	4	1	0	0	1
HMI-replay	350	0.13	VRT	2	0	0	27	2	0	304	1
HMI-masquerading	263	0.13	VRT	1	0	0	27	1	0	202	1
HMI-report-block	293	0.13	VRT	2	0	0	27	2	0	262	1
HMI-value-change	419	0.13	VRT	0	2*	0	27	2	0	369	1

by 69%). This experiment shows how important is to apply the automata-based reduction on the learned model.

D. Protocol IEC 104

The first experiment evaluates the accuracy and performance of the proposed automata-based approach on IEC 104 communication. We compare the results of DETANO with the original approach implemented in tool AUTANOM. The summary results are shown in Table VI.

In the first part of table, all datasets except HMI-MITM and RTU-MITM contain a normal traffic, so these datasets are used for method validation. We therefore learned the model from the initial part of the dataset traffic (∼10 % of the packets). Such learning sample is not as comprehensive as the one used in the second part of the table. Representatives of some kinds of normal traffic may be missing, which can lead to a higher number of false positives. To compensate for this, we selected a slightly higher threshold $\theta = 0.2$.

For the datasets in the second part of the table where each dataset contains some anomaly, a special traffic sample was used for the model learning, created with the learning in mind. In particular, for datasets from the VRT benchmark, we used a learning (normal) traffic consisting of 381,666 packets with a duration ∼2 days and for datasets from BUT we used a learning traffic consisting of 58,930 packets with a duration ∼3 days. For these datasets, we selected detection threshold $\theta = 0.13$.

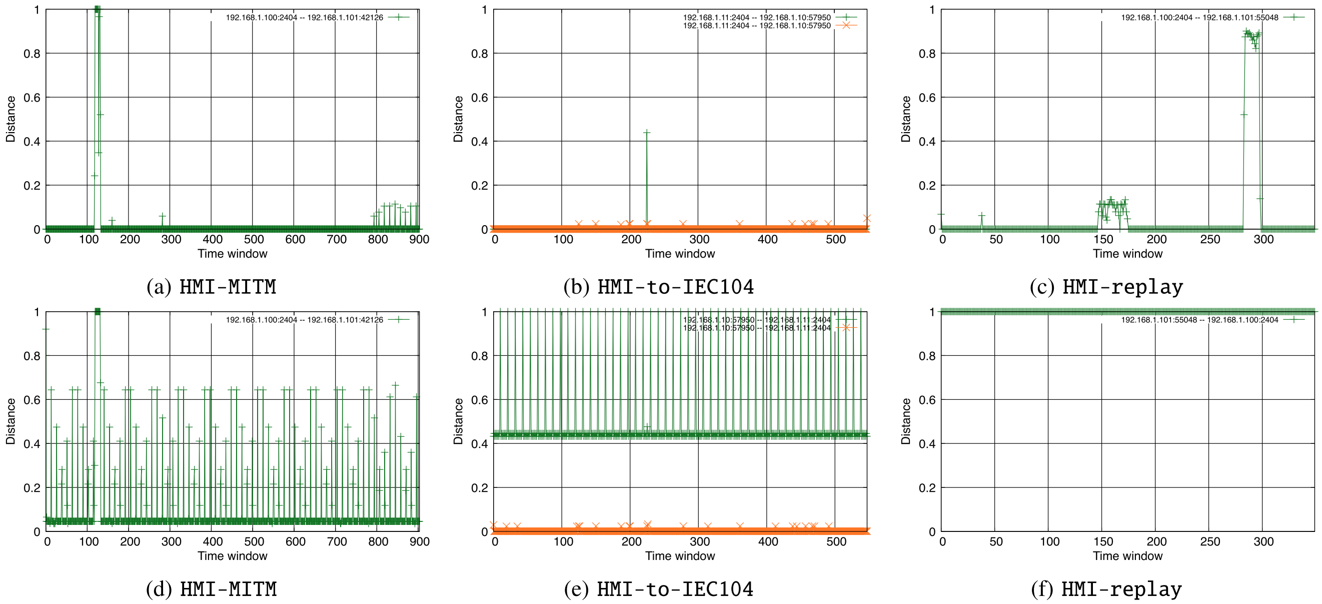
In all experiments from both parts of the table, removing identical model DPAs (model reduction with the threshold $\epsilon = 0.0$) was sufficient to keep the number of model DPAs small. The detection phase did not suffer from any performance issues and the model reduction was not needed.

a) *Improved Precision:* Table VI shows that the tool DETANO eliminates almost all false positives introduced by AUTANOM. Namely, 2,521 FPs in all datasets of AUTANOM were cut down to 3 FPs of DETANO. False positives were hence reduced by 99.9 %. An example of Euclid distance obtained from Eq. (2) for selected datasets comparing DETANO and AUTANOM is depicted in Fig. 12. It is apparent that DETANO smoothed most of the values corresponding to false positives while keeping peaks with anomalies, see, e.g., windows 118–132 in Fig. 12 (a) and (d).

Despite this FP reduction, the detection stays very precise. In particular, we missed only one attack, dos-attack, that could not be detected by our automata-based approach. All other missed attacks (marked by $*$) are beyond the scope of our approach because these attacks aim at features that are not trackable by DPA models. In particular, changes in IOA objects are not considered for the monitoring, thus, they are not represented by DPAs (cf. Section III-A).

The fact that AUTANOM detects these attacks is rather a side effect of its unacceptable high number of FPs, among which it accidentally hits the right windows with attacks. For instance, in case of HMI-MITM, AUTANOM marks 11 % of incoming windows as an anomaly, however, most of them are FPs. This number of FPs would require an administrator to resolve a false alarm approximately every hour which is not feasible. This is even more pronounced in the case of HMI-value-change, where almost *every* incoming window is marked as an anomaly.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.



(a) `HMI-MITM` (b) `HMI-to-IEC104` (c) `HMI-replay`

(d) `HMI-MITM` (e) `HMI-to-IEC104` (f) `HMI-replay`

Fig. 12. Comparison of the Euclid Distance Values Obtained From Eq. (2) for DETANO (a), (b), (c) and AUTANOM (d), (e), (f).

The high precision is not paid for by a performance decline. On all datasets (except of `10122018-104Mega` having the lowest number of windows where the processing of a single window takes 1 sec) we keep the time for processing a single window for a communication pair below 0.1 sec.

b) *Diagnostic Traces:* Our approach is able to provide useful diagnostic traces for detected anomalies allowing to track possible sources of problems. Below, we discuss a couple of interesting cases.

- *`connection-loss`:* For anomalies occurring within this dataset, we provide additional information about the most probable missing conversations $\mathcal{E}_{miss} = \{\langle 36, 3\rangle\}$. In this case the device stopped responding and hence the absence of this message was indeed marked as an anomaly.
- *`HMI-report-block`:* In this scenario, the attacker blocks and re-sends messages which yields into duplicated messages $\langle 3, 20\rangle$. We provide additional diagnostic traces containing $\mathcal{E}_{bad} = \{\langle 100, 6\rangle \langle 100, 7\rangle\langle 3, 20\rangle \quad \langle 3, 20\rangle\langle 5, 20\rangle \quad \cdots \langle 100, 10\rangle, \dots\}$ and $\mathcal{E}_{miss} = \{\langle 100, 6\rangle\langle 100, 7\rangle\langle 3, 20\rangle \quad \langle 5, 20\rangle \cdots \langle 100, 10\rangle\}$, based on which the administrator can reveal the problem.
- `HMI-masquerating`: In this scenario, the attacker generates cyclic requests yielding to a duplication of $\langle 100, 7\rangle$ messages. From our diagnostic trace $\mathcal{E}_{bad} = \{\langle 100, 6\rangle\langle 100, 7\rangle\langle 100, 7\rangle\langle 1, 20\rangle \cdots \langle 100, 10\rangle, \dots\}$ we are able to see duplicated messages $\langle 100, 7\rangle$.

It is evident from the scenarios above that the diagnostic traces generated by the detection system provide a valuable information for revealing the essence of an anomaly. Based on these traces, an administrator may quickly find an effective measure against a device failure or cyber attack.

TABLE VII
RESULTS OF DETANO ON MMS BENCHMARKS

Dataset	Windows	Benchmark	DETANO			
			Hit	Miss	FP	$\lvert\mathbb{A}\rvert$
`inside-substation`	114	ENCS	1	0	0	13
`gics-lost-connection`	94	GICS	2	0	0	36
`gics-modified`	94	GICS	2	0	0	36
`gics-scanning`	94	GICS	1	0	0	36
`gics-interrupt`	94	GICS	0	1	0	36

E. Protocol MMS

In the second experiment, we evaluated the accuracy and performance of DETANO on MMS benchmarks. We did not include AUTANOM as it was tailored for IEC 104 only. Since the MMS benchmarks contain separate learning parts, we set the same detection threshold $\theta = 0.13$ as for IEC 104 with the learning part. For the `ENCS` benchmark, the learning traffic had 13,043 packets with a duration ∼15 minutes, and for the `GICS` benchmark the learning traffic had 2,706 packets with a duration ∼ 1.5 hours. Regarding the model reduction, we again removed only identical DPAs. Since the MMS benchmarks capture only a shorter-term communication (less than 2 hours), we set the incoming window duration to 60 sec.

a) *Improved Precisions:* From Table VII we can see results concerning the anomaly detection in MMS communication involving a couple of scenarios. First observe, that DETANO is able to detect all anomalies except `gics-interrupt`. We are able to detect the anomalies occurring within this dataset with a smaller detection threshold $\theta = 0.09$. Further, we emphasize that our detection technique introduces no false positives. As well as in the case of the IEC 104 traffic, the processing of a single window for a communication pair took less than 0.1 sec.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

TABLE VIII
Overview and Comparison of Selected AD Methods. Attack Codes: CL - Connection Loss, DA - DoS Attack, IA - Injection Attack, SA - Scanning Attack, WA - Switching Attack, MM - Message Modification

Properties	Detano	[17]	[34]	[46]	[47]	[35]	[48]	[33]
Model	Automata	Periodogram	Automata	Bayesian	Autoregression	DTMC	Specification	Mean and Range
Construction	Auto	Auto	Auto	Auto	Auto	Auto	Manual	Auto
Protocols	IEC104, MMS	Modbus, DNP3	Modbus	Modbus	Modbus	Modbus, IEC104	Modbus	Modbus, IEC104, S7
System	Real+Virtual	None	Real	Testbed	Real+Testbed	Real	Real	Real+Virtual
Datasets	Public+Private	None	Private	Private	Private	Private	Private	Public+Private
Evaluation	FPR	None	FPR	FPR	FPR	FPR	None	FPR
Attacks	CL, IA, SA, WA	IG, DA	None	IA	MM	IA	None	DA, IA

b) *Diagnostic Traces:* Below, we discuss some interesting cases of providing diagnostic traces for MMS anomalies.

- `gics-lost-connection`: In this scenario, a pair of devices stops responding. Regarding a diagnostic trace, we report $\mathcal{E}_{miss} = \{\langle 0, 4\rangle\langle 1, 4\rangle\}$ (i.e, confirmed-request/response PDUs with read service), which is indeed a key missing conversation in the communication.
- `gics-scanning`: In this scenario, a malicious device starts scanning resources. Together, with an anomaly alert, we provide a diagnostic trace containing $\mathcal{E}_{bad} = \{\langle 0, 12\rangle\langle 1, 12\rangle, \dots\}$, i.e., the confirmed-req/resp PDUs with `GetNamedVariableListAttributes` service.

The diagnostic traces provide beneficial information about the anomalies. Operators can immediately infer the anomaly by inspecting a few provided conversations. For selected attacks (e.g., `gics-scanning`), the diagnostic traces directly reveal the intentions of an intruder.

F. Discussion

From the experimental evaluation it is evident that extended functionality implemented in Detano removes virtually all false positives while keeping the detection still very precise, which is a crucial objective for a real-world deployment. Another important outcome of the experiments is that the newly proposed multiple-model detection does not negatively affect the performance of the detection system. The number of models (automata) ranges from 1 to 123 (after model reduction with parameter $\epsilon = 0.0$). Moreover in the real-world deployment, if the number of models becomes too high, it is possible to apply more radical reductions for the price of detection precision. We also showed that the provided diagnostic traces are a very useful tool for a precise localization of the anomaly.

In the real-world setting may occur packet delays or retransmissions in the network. If this happens during the learning phase, we build a model with delayed or retransmitted packets and hence in the detection, our model is able to cope with such events without an anomaly alert. If the learning traffic does not contain this kind of messages, the detection system marks such conversations as suspicious since they are anomalous according to the model. (In fact, it depends on the frequency of such incidents. If their frequency is low, the retransmitted or delayed communication does not affect much a distribution describing overall communication and this small difference can be hidden behind the threshold.)

V. Comparison to Other Work

In this section, properties of the proposed method are compared with selected ICS anomaly detection techniques. Due to the fact that the details of implementations and datasets are not commonly available, the presented comparison is based solely on the information provided by the authors in their articles. The criteria take into account the representation of the model, the learning and detection process and the approach to assessment. Table VIII presents anomaly detection methods with respect to the following categories:

- *Model:* All considered methods propose a model of ICS communication that is based on characterizing a baseline. Different types of models are employed, most often statistical, automata, and probabilistic.
- *Construction:* Usually, the model is computed automatically from the sample data, but there are also methods that require a user to manually provide a system specification. The detection methods are specific to the model used. For simple statistical models, they determine if a tested value lies within the specified boundaries. In the case of automata, the acceptance of an input string is evaluated or the distance between automata is computed.
- *Protocols:* A method is applicable to a number of ICS protocols or is specific to a single one. Also, the authors present the method for various protocols depending on the availability of datasets and experimental environment. The most common are Modbus, IEC 104, DNP3, or S7.
- *System:* To evaluate the performance of detection methods the availability of datasets is essential. Many works use datasets acquired from either a real-world ICS system or emulated/virtual environment. A few of the presented works claim to use several ICS systems during the demonstration and evaluation.
- *Datasets:* Many methods were demonstrated and evaluated using private datasets. There are not so many publicly available datasets and if so most of them are generated in an experimental testbed rather than obtained from a real system. Most of available datasets are in form of CSV traces, some datasets contain full captured data.
- *Evaluation:* Most of the works evaluate the performance in terms of false-positive rate (FPR). Some of them discuss capabilities of the proposed method without providing quantitative evaluation.
- *Attacks:* Some authors also present the capabilities of the methods to detect various attacks. The types of attacks differ, but the most common are DoS attacks and

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

TABLE IX
SUMMARY OF ANOMALY DETECTION RESULTS. (✓ SUCCESSFUL DETECTION, ✗ NOT DETECTED, ○ PARTIALLY DETECTED)

Anomaly	Result	MITRE Technique Classification
Connection loss	✓	–
DoS attack	✗	`T0814`
Injection attack	✓	`T0855,T0856,T0881`
Rogue device	✓	`T0848`
Scanning attack	✓	`T0801,T0802,T0840,T0846,T0888`
Switching attack	✓	`T0838,T0858`
Replay attack	✓	`T0856`
MITM attack	○	`T0830`
Masquerading attack	✓	`T0831,T0856`
Message blocking	✓	`T0803,T0804,T0878`
Message modification	○	`T0831,T0832,T0836`

injection attacks as these significantly modify the traffic characteristics.

The mainstream techniques employed for ICS anomaly detection found in the literature are statistical methods, automata, and probabilistic models. The presented experiments demonstrate that our DFA approach can detect most of the network attack types considered in the related works (see Table VIII, row Attacks). One exception is the message modification (MM) attack that is detectable by the autoregression method [47]. This limitation is because we do not analyze message payloads. If message modification does not modify the message type nor introduces new messages in the communication, it is not detectable by our method. Also, the DoS attack, easily detectable by statistical methods, is not identified if the individual requests obey the expected communication pattern. The overview of attacks detectable by our system with corresponding MITRE classification is given in Table IX.

As mentioned above, DoS attack is not detectable by the automata-detection because a DPA models only relative frequency of the learned conversations, thus, the same DPA is created for an ICS sequence that occurs ten times in the time window (normal behavior) or thousand times (DoS attack). Nevertheless, DoS attack can be easily detected by a simple statistical analysis. Table IX also lists a few partial results marked with (○) for the MITM attack and message modification. Since our system does not analyze packet payload, it is not able to detect packets with modified values in the payload. However, such values usually invoke a new reaction of the ICS system that is trackable by our detection tool. This was demonstrated by our experiments with IEC 104 (Table VI, datasets `HMI-MITM`, `RTU-MITM`) and with MMS (Table VII, dataset `gics-modified`) where unexpected IEC 104 or MMS response message were detected. We denote this result as partial (○) which means that we did not capture the original attack but a response for the attack.

VI. CONCLUSION

Industrial control networks use segmentation and perimeter protection for the critical parts of the system, which was rendered insufficient due to new attack vectors identified and successfully exploited by an intruder. Once the intruder gets into a protected Operation Technology (OT) segment, he/she may cause a serious damage to the controlled physical environment. This paper described a method for monitoring smart grid networks and automatic detection of deviations from the normal communication. A detected anomaly either corresponds to an attack or malfunctioning device, both of which require immediate operator's attention.

The proposed method constructs probabilistic automata from the observed ICS communication that represent a language by which ICS devices in the smart grid network usually talk. The method employs application-level information in order to model the ICS communication accurately. Potential attackers need to follow the exact steps of the system communication to go undetected, which significantly limits their movements. Compared to other approaches commonly based on statistical, automata, and probabilistic models, the method detects a wider range of attack types (see Table IX). In addition, our method provides a trace for the identified anomaly, which is difficult to obtain by machine learning approaches, for example. These diagnostic traces help an administrator further investigate an incident and determine the root cause of a system failure.

An essential requirement for practical deployment of anomaly detection is an almost zero false-positives rate. The previous experiments demonstrated that the FP rate of the proposed method is very low or, in many cases, even zero. Another practical requirement is overall complexity and conservative resource demands. Using large datasets of ICS communication, we proved that the computed model is very compact even for millions of observed communication flows. Also, model computation and anomaly detection algorithms are computationally feasible but this task is done offline during the training phase only.

The presented method was implemented and integrated in the network monitoring and anomaly detection tool[8] for further testing in real-world environments.

REFERENCES

[1] E. D. Knapp and J. T. Langill, *Industrial Network Security. Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Waltham, MA, USA: Syngress, 2015.

[2] T. Miller, A. Staves, S. Maesschalck, M. Sturdee, and B. Green, "Looking back to look forward: Lessons learnt from cyber-attacks on industrial control systems," *Int. J. Crit. Infrastruct. Protect.*, vol. 35, Dec. 2021, Art. no. 100464.

[3] K. E. Hemsley and D. R. E. Fisher, "History of industrial control system cyber incidents, INL/CON-18-44411-Revision-2." Dec. 2018. [Online]. Available: https://www.osti.gov/biblio/1505628

[4] "Guidelines for smart grid cybersecurity," NIST, Gaithersburg, MD, USA, Rep. NISTIR-7628r1, 2014.

[5] K. Stouffer, V. Pillitteri, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Rep. NIST-SP-800-82r2, 2015.

[6] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the Ukrainian power grid. defense use case," Electr. Inf. Sharing Anal. Center, Washington, DC, USA, Rep., Mar. 2016.

[7] A. Cherepanov, "Win32/Industroyer. A new threat for industrial control systems," ESET, Bratislava, Slovakia, Rep., Jun. 2017.

[8] "CrashOverride. Analysis of the threat of electric grid operations," Dragos, Inc., Hanover, MD, USA, Rep., Jun. 2017.

[8] See Flowmon ADS [Feb 2022].

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

[9] B. Johnson, D. Caban, M. Krotofil, D. Scali, N. Brubaker, and C. Glyer, "Attackers deploy new ICS attack framework 'TRITON' and cause operational disruption to critical infrastructure," *Threat Res. Blog*, vol. 14, 2017.

[10] M. Giles, *Triton Is the World's Most Murderous Malware, and It's Spreading*, MIT Technol. Rev., Cambridge, MA, USA, 2019.

[11] N. Nelson, *The Impact of Dragonfly Malware on Industrial Control Systems*, SANS Inst., Bethesda, MD, USA, Jan. 2016.

[12] J. L. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell, "A quantitative evaluation of the target selection of havex ICS malware plugin," in *Proc. Ind. Control Syst. Security (ICSS) Workshop*, 2015, pp. 1–5.

[13] E. Kovacs. "PLC worms can pose serious threat to industrial networks." Security Week. May 2016. [Online]. Available: https://www.securityweek.com/plc-worms-can-pose-serious-threat-industrial-networks

[14] P. Matoušek, O. Ryšavý, M. Grégr, and V. Havlena, "Flow based monitoring of ICS communication in the smart grid," *J. Inf. Security Appl.*, vol. 54, Oct. 2020, Art. no. 102535.

[15] P. Matoušek, V. Havlena, and L. Holík, "Efficient Modelling of ICS communication for anomaly detection using probabilistic automata," in *Proc. IEEE/IFIP Int. Symp. Integr. Netw. Manage. (IM)*, 2021, pp. 81–89.

[16] R. R. R. Barbosa, R. Sadre, and A. Pras, "A first look into SCADA network traffic," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 518–521.

[17] R. R. R. Barbosa, R. Sadre, and A. Pras, "Towards periodicity based anomaly detection in SCADA networks," in *Proc. IEEE 17th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2012, pp. 1–4.

[18] A. Valdes and S. Cheung, "Communication pattern anomaly detection in process control systems," in *Proc. IEEE Conf. Technol. Homeland Security*, May 2009, pp. 22–29.

[19] S. S. Jung, D. Formby, C. Day, and R. Beyah, "A first look at machine-to-machine power grid network traffic," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Nov. 2014, pp. 884–889.

[20] C. De La Higuera, *Grammatical Inference: Learning Automata and Grammars*. New York, NY, USA: Cambridge Univ. Press, 2010.

[21] O. Alexander, M. Belisle, and J. Steele, "MITRE ATT&CK for industrial control systems: Design and philosophy," MITRE, McLean, VA, USA, Rep. MPO1055863, Mar. 2020.

[22] S. V. B. Rakas, M. D. Stojanović, and J. D. Marković-Petrović, "A review of research work on network-based SCADA intrusion detection systems," *IEEE Access*, vol. 8, pp. 93083–93108, 2020.

[23] A. S. Musleh, G. Chen, and Z. Y. Dong, "A survey on the detection algorithms for false data injection attacks in smart grids," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2218–2234, May 2020.

[24] X. Chen, S. Hu, Y. Li, D. Yue, C. Dou, and L. Ding, "Co-estimation of state and FDI attacks and attack compensation control for multi-area load frequency control systems under FDI and DoS attacks," *IEEE Trans. Smart Grid*, vol. 13, no. 3, pp. 2357–2368, May 2022.

[25] Z. Zheng, Y. Yang, X. Niu, H.-N. Dai, and Y. Zhou, "Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1606–1615, Apr. 2018.

[26] Y. Zhao, W. Yao, C.-K. Zhang, X.-C. Shangguan, L. Jiang, and J. Wen, "Quantifying resilience of wide-area damping control against cyber attack based on switching system theory," *IEEE Trans. Smart Grid*, vol. 13, no. 3, pp. 2331–2343, May 2022.

[27] S. Lakshminarayana, E. V. Belmega, and H. V. Poor, "Moving-target defense against cyber-physical attacks in power grids via game theory," *IEEE Trans. Smart Grid*, vol. 12, no. 6, pp. 5244–5257, Nov. 2021.

[28] M. Altaha and S. Hong, "Anomaly detection for SCADA system security based on unsupervised learning and function codes analysis in the DNP3 protocol," *Electronics*, vol. 11, no. 14, p. 2184, 2022.

[29] L. Yang, Y. Zhai, Y. Zhang, Y. Zhao, Z. Li, and T. Xu, "A new methodology for anomaly detection of attacks in IEC 61850-based substation system," *J. Inf. Security Appl.*, vol. 68, Aug. 2022, Art. no. 103262.

[30] S. Ashraf, M. Shawon, H. M. Khalid, and S. M. Muyeen, "Denial-of-service attack on IEC 61850-based substation automation system: A crucial cyber threat towards smart substation pathways," *Sensors*, vol. 21, no. 19, p. 6415, 2021.

[31] P. Matoušek, O. Ryšavý, and M. Grégr, "Increasing visibility of IEC 104 communication in the smart grid," in *Proc. 6th Int. Symp. ICS SCADA Cyber Security Res.*, 2019, pp. 21–30.

[32] C.-Y. Lin and S. Nadjm-Tehrani, "Understanding IEC-60870-5-104 traffic patterns in SCADA networks," in *Proc. 4th ACM Workshop Cyber-Phys. Syst. Security*, 2018, pp. 51–60.

[33] C.-Y. Lin, S. Nadjm-Tehrani, and M. Asplund, "Timing-based anomaly detection in SCADA networks," in *Proc. Int. Conf. Crit. Inf. Infrastruct. Security*, 2017, pp. 48–59.

[34] N. Goldenberg and A. Wool, "Accurate modeling of modbus/TCP for intrusion detection in SCADA systems," *Int. J. Crit. Infrastruct. Protect.*, vol. 6, no. 2, pp. 63–75, 2013.

[35] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proc. 1st ACM Workshop Cyber-Phys. Syst. Security*, 2015, pp. 13–24.

[36] M. Caselli, E. Zambon, J. Petit, and F. Kargl, "Modeling message sequences for intrusion detection in industrial control systems," in *Critical Infrastructure Protection IX*. Cham, Switzerland: Springer, 2015, pp. 49–71.

[37] Y. Wang, Z. Zhang, D. D. Yao, B. Qu, and L. Guo, "Inferring protocol state machine from network traces: A probabilistic approach," in *Applied Cryptography and Network Security*, J. Lopez and G. Tsudik, Eds. Heidelberg, Germany: Springer, 2011, pp. 1–18.

[38] T. Krueger, H. Gascon, N. Krämer, and K. Rieck, "Learning stateful models for network honeypots," in *Proc. 5th ACM Workshop Security Artif. Intell.*, New York, NY, USA, 2012, pp. 37–48.

[39] O. Ryšavý and P. Matoušek, "A network traffic processing library for ICS anomaly detection," in *Proc. 7th Conf. Eng. Comput. Based Syst.*, 2021, p. 22.

[40] E. Vidal, F. Thollard, C. De La Higuera, F. Casacuberta, and R. C. Carrasco, "Probabilistic finite-state machines—Part I," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 7, pp. 1013–1025, Jul. 2005.

[41] P. Matoušek, O. Ryšavý, and P. Grofčík. "ICS Dataset for smart grid anomaly detection." 2022. [Online]. Available: https://dx.doi.org/10.21227/1trw-n685

[42] C.-Y. Lin, A. Fundin, E. Westring, T. Gustafsson, and S. Nadim-Tehrani, "RICSel21 data collection: Attacks in a virtual power network," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, 2021, pp. 201–206.

[43] M. Kabir-Querrec, S. Mocanu, J.-M. Thiriet, and E. Savary, "A test bed dedicated to the study of vulnerabilities in IEC 61850 power utility automation networks," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, 2016, pp. 1–4.

[44] A. D. Pinto and Y. Stavrou. "Your guide to the MITRE ATT&CK framework for ICS." Aug. 2020. [Online]. Available: https://www.nozominetworks.com/blog/your-guide-to-the-mitre-attack-framework-for-ics/

[45] M. J. Assante and R. M. Lee, "The industrial control system cyber kill chain," SANS Inst., Bethesda, MD, USA, Rep., Oct. 2015.

[46] M.-K. Yoon and G. F. Ciocarlie, "Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems," in *Proc. NDSS*, 2014, pp. 1–10.

[47] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Semantic security monitoring for industrial processes," in *Proc. 30th Annu. Comput. Security Appl. Conf.*, 2014, pp. 126–135.

[48] M. Faisal, A. A. Cardenas, and A. Wool, "Modeling modbus TCP for intrusion detection," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2016, pp. 386–390.

Vojtěch Havlena is a Researcher with the Faculty of Information Technology, Brno University of Technology, Czech Republic. His research focus includes formal methods, automata theory, and logics.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.

Petr Matoušek is an Associate Professor and a Researcher with the Brno University of Technology, Czech Republic. He was a Principle Investigator and a Research Team Member of many national and international research projects. His research focus includes network monitoring, cyber security, industrial communication, and Internet of Things. He is a member of IEEE COMMUNICATION SOCIETY, European Alliance for Innovations, and European Association for Education in Electrical and Information Engineering.

Ondřej Ryšavý is an Associate Professor with the Faculty of Information Technology, Brno University of Technology, Czech Republic. He has participated in national and international research and development projects and collaboration with the industry. His research focuses on computer networking, cyber security, digital forensics, and incident response.

Lukáš Holík is an Associate Professor with the Faculty of Information Technology, Brno University of Technology, Czech Republic. He was mainly focusing on theoretical aspects of this work. His research includes formal methods, automated reasoning, verification, theory and applications of automata, and logic and decision procedures.

Authorized licensed use limited to: Brno University of Technology. Downloaded on April 21,2023 at 20:10:36 UTC from IEEE Xplore. Restrictions apply.