Temporal Phase Shifts in SCADA Networks

Chen Markman
School of Electrical Engineering
Tel Aviv University
Ramat Aviv, Israel
chenmark@post.tau.ac.il

Avishai Wool
School of Electrical Engineering
Tel Aviv University
Ramat Aviv, Israel
yash@acm.org

Alvaro A. Cardenas
Department of Computer Science
University of Texas at Dallas
Richardson, Texas, USA
alvaro.cardenas@utdallas.edu

ABSTRACT

In Industrial Control Systems (ICS/SCADA), machine to machine data traffic is highly periodic. Previous work showed that in many cases, it is possible to create an automata-based model of the traffic between each individual Programmable Logic Controller (PLC) and the SCADA server, and to use the model to detect anomalies in the traffic. When testing the validity of previous models, we noticed that overall, the models have difficulty in dealing with communication patterns that change over time. In this paper we show that in many cases the traffic exhibits phases in time, where each phase has a unique pattern, and the transition between the different phases is rather sharp. We suggest a method to automatically detect traffic phase shifts, and a new anomaly detection model that incorporates multiple phases of the traffic. Furthermore we present a new sampling mechanism for training set assembly, which enables the model to learn all phases during the training stage with lower complexity. The model presented has similar accuracy and much less permissiveness compared to the previous general Deterministic Finite Automata (DFA) model. Moreover, the model can provide the operator with information about the state of the controlled process at any given time, as seen in the traffic phases.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Computer systems organization → Embedded and cyber-physical systems**;

ACM Reference Format:
Chen Markman, Avishai Wool, and Alvaro A. Cardenas. 2018. Temporal Phase Shifts in SCADA Networks. In *Workshop on Cyber-Physical Systems Security & Privacy (CPS-SPC '18), October 19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3264888.3264898

1 INTRODUCTION

1.1 Background

Industrial Control Systems (ICS) are used for monitoring and controlling numerous industrial systems and processes such as chemical plants, electric power generation, waste-water treatment facilities, etc. ICS is a general term that encompasses several types of control systems, including Distributed Control Systems (DCS), Supervisory Control And Data Acquisition (SCADA) systems, and other control system configurations [26].

ICS were originally built on the premise that all the operating entities would be legitimate, and follow the protocols of the system. As recent attacks have shown, it is no longer safe to assume that all devices in an ICS are trusted, therefore, deploying an anomaly detection system in an ICS network is an important defensive measure. The most famous example was the attack on an Iranian nuclear facility in 2010 (Stuxnet) to sabotage centrifuges at the Natanz uranium enrichment plant [11, 18]. Another notable example is the 2015 attack on the Ukrainian power grid [5], causing major service outages to customers.

1.2 Related work

Anomaly Detection: Surveys of techniques related to learning and detection of anomalies in critical control systems can be found in [1, 27].

Different kinds of anomaly-based IDS models have been suggested for SCADA systems [2, 8, 9, 13, 15, 28, 29]. Several of these models, however, have difficulties in explaining the reasoning behind each alert. Sommer and Paxson [24] argue that one of the reasons for the slow adoption rate of learning-based anomaly detection systems is that they lack the ability to bypass the "semantic gap": The system "understands" that an abnormal activity has occurred, but it cannot help the operator differentiate between an abnormal activity and an attack. One of our goals is to produce an anomaly detection model that can inform operators why the activity is being reported and why it is important.

Automata-based models The periodicity of industrial control networks has been documented in several studies [3, 4, 17]. This periodicity can be captured by automata-based models, where requests and responses of industrial control networks are modeled as jumps between recurrent states.

In one of the first papers on the topic, Goldenberg & Wool [14] developed a model-based approach using a Deterministic Finite Automata (DFA) to represent the cyclic nature of the commands exchanged in Modbus traffic.

Caselli et al. [6] proposed a probabilistic Discrete-Time Markov Chain (DTMC) model to capture sequences of SCADA messages. Based on data from three different Dutch utilities, the authors found that only 35%-75% of the possible transitions in the DTMC were observed. This strengthens the observations of a substantial sequentiality in SCADA communications [3, 14, 16]. However, unlike [14, 16] they did not observe clear cyclic message patterns.

CPS-SPC '18, October 19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5992-4/18/10…$15.00
https://doi.org/10.1145/3264888.3264898

1.3 Contributions

Our starting point is the work of Markman et al. [20]. Our first contribution is a new method for "fingerprinting" traffic patterns in a given period of time. This enables us to introduce a measure of similarity between traffic patterns at different points in time.

Our second contribution is demonstrating that the traffic patterns in the data channels have phases—there are stable traffic patterns followed by sudden changes into different, but stable, traffic patterns.

Furthermore, we introduce a method to automatically detect these traffic phase shifts. Our algorithm is based on clustering the different time periods using our measure of similarity. This enables us to describe the traffic in a channel as a sequence of phases in time—potentially bridging (some of) the semantic gap that exists in anomaly detection models.

Our fourth contribution is the introduction of an anomaly detection model that is phase aware, and the introduction of a sampling method to assure that the training set contains samples from the different phases in the channel.

Finally, we introduce a permissiveness measure for the new model, developed by adjusting the measure of [20].

We conducted an extensive evaluation of our new model on a large water facility data corpus [10], and our model, together with the sampling mechanism we suggested, improves the accuracy of previous models, while lowering the measure of permissiveness: i.e., the model has fewer false alarms, and is more specific. A full technical report of our results can be found in [21].

2 PRELIMINARIES

2.1 The GW model

The GW model [14] was developed and tested on Modbus traffic. A typical Modbus packet carries information about the *message type*, the *function code* specifying the service (e.g., read or write), and the *memory address range* of data items. After the PLC processes the request, it sends a response back to the HMI.

In the GW model, the key assumption is that traffic is *periodic*, therefore, each HMI-PLC channel is modeled by a Mealy Deterministic Finite Automaton (DFA). The DFA for Modbus has the following characteristics: (a) A symbol is defined as a concatenation of the *message type, function code*, and *address range*;(b) A state is defined for each message in the periodic traffic pattern. In the learning stage, a Mealy DFA is built for each HMI-PLC channel based on the training set. In the enforcement stage, the system triggers anomalies when the traffic is not recognized by its DFA. The model includes 3 types of anomalies: "unknown" symbol, not seen during the training stage, "miss" for symbols that appear out of order, and "retransmit" symbols.

2.2 The Burst-DFA model

The Burst-DFA model [20] was developed by Markman et al. The research was done by analyzing a corpus of Modbus traffic recorded at a water treatment plant in the U.S, which was previously found to be poorly modeled by cyclic-DFA models [10]. The research found evidence of parallel TCP connections between the HMI and PLCs, using different ports, and therefore the channel is defined by the tuple (HMI IP, PLC IP, Unit-ID, PLC's port). A major finding of [20] is that for each channel, the traffic exhibits bursty behavior—the HMI sends queries in bursts with defined construction, and with a relatively long time difference between consecutive bursts. Further, the research showed that the bursts have semantic meaning—the order within a burst depends on the messages.

Based on these observations a new model was suggested, which for each channel comprises a DFA that matches all the bursts of that channel, including their beginning and ending. For each channel, the burst-DFA learned the normal bursts one expect to see. The training stage was done by dividing the data into channels, splitting each channel packet stream into bursts of packets, and building a directed graph in the form of an adjacency matrix. In the enforcement stage, the model uses the burst-DFA to evaluate each data burst as it arrives, and ranks each query packet according to its position in the burst. The Burst-DFA model uses the 3 anomaly indicators from the GW model (unknown, miss, retransmit), and adds two extra messages "bad-beginning" and "bad-ending" to indicate that a burst doesn't start or doesn't end with a symbol that was previously seen in these positions.

The study introduced a metric to evaluate the permissiveness of the model—how strict or how general the model is. We discuss this metric in section 6.

The burst-DFA model was evaluated on the water treatment data corpus. The Burst-DFA model successfully explains between 95% to 99% of the packets in the data-corpus, when the training set includes 50% of the data.

3 THE WATER TREATMENT PLANT DATA CORPUS AND FORMAL DEFINITIONS

3.1 Overview

We used a one-day recording from a water treatment plant in the U.S. Since Modbus is a Master/Slave protocol, we concentrate on modeling only the query packets by the HMI. Modeling only the queries in a Master/Slave protocol is possible because for each query, there is a single response packet whose meta-data is fully determined by the query. Therefore modeling the responses does not add information in a model that focuses only on the meta-data. The recording duration was 24Hrs, 3 minutes and 19 seconds, it contains 68, 886, 147 total packets, the packet rate was 795 packets per second, the packet loss was 1.8%, and there were 99 different IP addresses.

3.1.1 Channel Separation and Identification. In this research we follow the definition of a channel from [20], and so we use the 4-tuple (Master IP, Slave IP, Unit ID, Slave Port) to define a channel. By "Slave Port" we mean the source port that the HMI's TCP connection uses when sending a query—note that the "Master Port" is always 502 in Modbus. The Unit ID field is used to address multiple Modbus slaves at a single IP address. We found 935 channels that exchange more than 500 packets (covering 99.29% of the packet capture). In the rest of the study, the numbering of the channels is arbitrary.

3.1.2 Input Symbols and States. We follow the definitions of the symbols from the GW model and the Burst-DFA model:
The states that are reached after a query message are called Q-states. Respectively, states that are reached after response messages are

called R-states. As mentioned, we have chosen to only model the sequence of queries in each channel. A Modbus query consists of the following fields: Transaction Identifier ($T.ID$), Function Code (FC), Reference Number (RN), and bit/word count ($Count$). We define a symbol in the alphabet as a 3-tuple ($FC, RN, Count$). We say that a symbol is a known-symbol if it appears in the training set of the particular channel, and an unknown-symbol otherwise. For each symbol s_i, we define a state S_i, as the DFA state following the occurrence of the symbol. A formal definition of a Deterministic Finite Automata (DFA) can be found in [20, 21].

4 TRAFFIC PHASES

When evaluating the performance of the Burst-DFA, we noticed that the accuracy deteriorates over time—on average the model describes the traffic that came right after the training data better than later traffic. Deeper inspection into the accuracy of the model on different channels showed abrupt changes in accuracy over time—sudden decline in accuracy, followed by returns to higher values. This indicates that there are "phases" in the traffic—at different times, the traffic follows different rules and regularities. A complete analysis of the performance in time can be found at the technical report [21]. After observing the existence of traffic phases by looking at the accuracy level of the Burst-DFA model over time, we turn our attention to detecting traffic phases directly, without using the result of the Burst-DFA model as a proxy. We define "traffic phases" as parts of the packet stream in a specific channel that exhibit a distinct pattern or a distinct set of possible packets.

4.1 Using Adjacency Matrices to identify phases in the data

The Burst-FDA model [20] builds an adjacency matrix for each channel, incorporating the different possible bursts of data—the possible set of packets and the possible order of the packets. An adjacency matrix is a description of a Deterministic Finite Automata (DFA), in which the nodes are the rows and columns of the adjacency matrix (unique packets), and the values of the matrix are the number of transitions between the nodes. In the next sections we describe the process of building the adjacency matrices, and introduce a measure of similarity between adjacency matrices, that allows us to divide the network traffic stream into different segments, similar to each other. If two adjacency matrices are similar, they describe similar traffic patterns, and therefore the traffic probably belongs to the same phase.

4.1.1 Building the adjacency matrix. We begin by separating the traffic into different channels as described in section 2. We then divide the packet stream in each channel to bursts of data, as described in [20], which are the input to the algorithm. The algorithm for building the adjacency matrix from a stream of bursts can be found in [20, 21].

4.1.2 Computing the similarity between adjacency matrices. Given two adjacency matrices, we wish to calculate a score measuring their similarity. To do so we reshape each adjacency matrix into a vector shape, and normalize each vector to unit length by dividing it by its magnitude. We then use the Euclidean distance as a measure of similarity—a small distance means similar adjacency matrices.

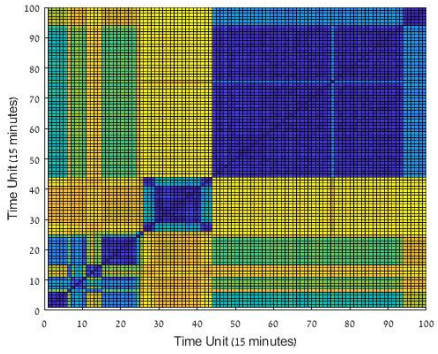

Figure 1: Channel 130 distance between adjacency matrices from different times

4.1.3 Using adjacency matrices to describe the data. We use the idea of the adjacency matrices as descriptors of the data stream in order to find the different phases in the traffic. We use the following procedure: 1) Divide the data into channels. 2) For each channel, separate the data stream into bursts of packets as in [20]. 3) Separate the list of bursts into 100 equal parts (arbitrary number). 4) Using the algorithm from section 4.1.1 to build an adjacency matrix of size sXs for each part, when s is the number of unique symbols in the channel. 5) Vectorize and normalize each matrix. 6) Compute the Euclidean distance between each pair of vectors.

4.2 Specific channel examples

After calculating the chosen similarity measure between each pair of the 100 adjacency matrices, we plotted the distance matrix for the channels and inspected them. For example, Figure 1 shows the distance matrices between the 100 parts of channels 130.

The block formation in Figure 1 demonstrates the distinct phases in channel 130—for example between the 45^{th} time unit to the 95^{th} time unit. The example shows that the phases can last for several hours, and that some phases appear only late in the recording.

5 ANALYZING THE TRAFFIC PHASES

When looking at Figure 1, we can visually detect the different phases. We would like to detect the phases automatically in order to check whether or not the traffic in a certain channel is divided into phases, and in order to build a traffic model. We want to segment the time series into different phases; in order to do that we organize the time parts into clusters based on the similarity between their adjacency matrices. To achieve this time series segmentation, we use k-means clustering [19]. While there are many different methods of choosing the number of clusters k, there is no definitive answer to this problem [7, 22]. We chose to use the silhouette method of determining k [23]. More details can be found in the technical report [21].

Example: To demonstrate the results of the clustering algorithm, we continue with channel 130 we discussed previously. By looking at Figure 1 we observed that there are around 7 phases, so in this example we manually chose k=8. We can see in Figure 2 that the algorithm recognized the same phase we observed between the 45-95 time units.

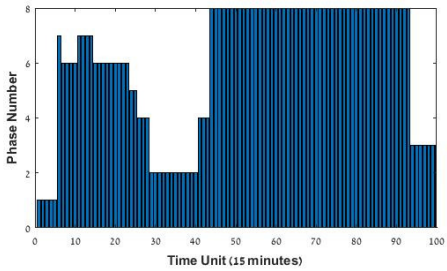

Figure 2: k-means cluster analysis of channel 130: the X axis is the time slot t, the Y axis is the number of the cluster that is assigned to the adjacency matrix at time t.

5.1 Phase shift analysis

Once we have a clustering method to identify the phases, we can study the behavior of the phases in the different channels. We found that some of the channels exhibit clear multi-phase behavior as seen in channel 130 described above (Figure 1), but other channels do not exhibit any such behavior—the distance matrix doesn't look like a block matrix, and there are no continuous phase patterns.

After running the clustering algorithm on all channels, we counted the number of phase shifts along the recording in order to describe the overall behavior of the channels. Intuitively, channels with only a few phase shifts exhibit phases, and channels with many phase shifts exhibit no such behavior. We found that 44% of the channels have 10 or less phase shifts—those are channels that exhibit some kind of multi-phase behavior. Conversely, 38% of the channels have more than 25 phase shifts, i.e., our analysis showed no multi-phase behavior. The number of phase shifts histogram can be found in the technical report [21]. Possibly some of the channels we found to have many phase shifts may still have phases, but our model didn't fit these channels well enough (maybe other choice of hyper-parameters or model would disclose multi-phase behavior).

We argue that automatic phase-shift detection may be of value beyond anomaly detection in the traffic. We believe that a traffic phase lasting many hours probably has semantic meaning that is correlated with a phase in the controlled process. As such, it may be labeled by the process engineer (e.g., as "Chlorinating"/"Mixing" etc.) during the training period, and displayed visually during the model enforcement. This type of semantic labeling is similar to the approach of [12] and may assist in reducing the semantic gap [25].

To illustrate the connection between the phase shifts and the controlled process, we analyzed a particular phase shift in channel 130 at the Modbus packets level. As can be seen in Figure 1, there is a phase shift after about 25% of the recording—about 6 hours into the recording. When looking at the bursts assignment into phases (as described in section 6), a new set of traffic patterns appears at that time. The set of patterns includes a new query that was previously unseen—a request to read the 2 bytes from a new register range (Modbus Reference Number)—1186. This example implies that the phase shifts are actually connected to a change in the operation of the controlled system, and hence has the potential to bridge some of the semantic gap.

6 A PHASE AWARE ANOMALY DETECTION MODEL

After witnessing the existence of traffic phases, we suggest a model that incorporates the phase detection capability, to provide high accuracy while limiting the permissiveness of the model. Our model creates a few sub-models during the training stage, according to the different traffic phases, and checks to see if the data fits one of the sub-models during the enforcement stage.

6.1 Training set assembly via burst-based sampling

As seen in Figure 1, traffic phases may only appear after a long time from the beginning of the recording—more than 10 hours in this example. It is important to include samples from all traffic phases in the training set in order to learn all possible patterns and to have an high accuracy model. In the Burst-DFA model [20] this resulted in taking the training set to be 50% of the data, which resulted in an accurate but highly permissive model. We suggest another option to assemble the training set—burst-based sampling. The main idea is to collect every n^{th} burst of data and include it into the training set, while skipping the rest. This way we can have a long training period while reducing the amount of traffic we need to gather, and still include all traffic phases. It is important to emphasize that we sample bursts of data, and not individual packets, in order to maintain the use of the internal structure of the bursts. We use the sampling mechanism to demonstrate the importance of the inclusion of all phases in the training set. We train the model on a sampled set of $\frac{1}{n}$ of the bursts, and test the model on the remaining $(1 - \frac{1}{n})$ of the bursts.

6.2 The training stage

In the training stage we begin forming a training set of bursts according to the steps discussed in section 5 and the burst-based sampling method. By repeating the steps from sections 5 and 6 on the training set we form 100 adjacency matrices, each labeled by a number from 1 to k, according to the cluster it is in. We now combine all of the adjacency matrices in each cluster using logical OR (we ignore the frequency of transitions), and we remain with k adjacency matrices (each a union of adjacency matrices from a cluster) representing the model—k DFA's in total. The model is different from the Burst-DFA model [20], that was made of one big adjacency matrix, formed as a union of the 100 adjacency matrices from our model.

6.3 The enforcement stage

Given k adjacency matrices, representing the k DFA's for a channel, we can compare the channel's traffic to the model, and flag anomalies in the enforcement stage. Similarly to the model presented in [20], we evaluate finite bursts—we move through the adjacency matrices from the starting state q_0 of each burst, and ensure we reach q_{end} at the end of the burst. Unlike [20], each burst of data is compared to all k DFA's from the training stage, in order to check for anomalies, and in order to determine to which traffic phase the burst belongs.

6.3.1 Single burst phase assignment. Each burst of data in the test stage is compared to all k DFAs from the training stage—for each one a vector of the counters of the different categories is formed *(Normal, Miss, Unknown, Retransmit, Wrong-Beginning, Wrong-Ending)*. We choose to assign the burst to one of the k phases by choosing the adjacency matrix that minimizes the Unknowns for the burst. If more than one adjacency matrix gives the same minimal number of unknowns, we take the one that results in the lowest number of overall anomalies *(Miss, Retransmit, Wrong-Beginning, Wrong-Ending)*. The output for each burst is the phase it is assigned to, and the vector of transition function categories.

6.3.2 Enforcement stage output. The enforcement stage of a particular channel is made by evaluating all of the bursts in the test set according to the k DFA's provided by the training stage. For each channel, the result of the enforcement stage is a vector of counters of the different transition function categories: Normal, Miss, Unknown, Retransmit, Wrong-Beginning, Wrong-Ending. The values are the sum of the counters for all of the bursts in the channel's test set. The technical report [21] brings a summary of the implementation of the enforcement process.

6.4 Analyzing the Permissiveness of the Model

It is important to understand how constrained or how permissive our model is. In the extreme, a channel with a single burst pattern will generate a single linear DFA with only one path from q_0 to q_{end}—a very constrained model. Conversely, if all s symbols are observed in every one of the b positions in the burst, then the model will allow all s^b paths through it—a permissive model. In [20] we introduced the R_{perm} measure to describe the level of permissiveness of the model. The permissiveness measure is a normalized ratio between the number of paths the model "allows", and the number of potential paths allowed through the most permissive model. We developed this measure to incorporate the structure of the new model, by introducing a way to calculate the number of unique paths in a set of graphs. Details can be found in the full technical report [21].

7 MODEL RESULTS

We introduced two main concepts in our model: 1. Burst-based sampling the training data (section 6). 2. Using k DFAs to evaluate the bursts in each channel. In this section we present the results in comparison with the Burst-DFA model [20], and we show the results with and without the sampling mechanism. The k-phase model permits fewer transitions between states compared to the Bursrt-DFA model, so given the same training set, we expect the accuracy of the model to be slightly worse than that of the Burst-DFA model. However, we expect the new model to be less permissive. Having said that, the sampling mechanism should improve the accuracy of the model, so a success will be a combination of the sampling mechanism and the new model thats result in better accuracy and less permissiveness than the Burst-DFA model without sampling. We demonstrate the results using a training set that is 33% of the data set. We describe the accuracy of the model by the percentage of normal queries, and by the percentage of queries that are either normal, miss, or retransmit. In addition, in the technical report [21] we checked the Wrong-Beginning and Wrong-Ending ratio (out of all bursts), to understand how well we model the structure of the bursts.

Table 1: Results Summary

model\accuracy ratio	normal ratio	normal+miss+ retransmit ratio
k-phase	88.2%	99.7%
k-phase+sampling	98.9%	99.99%
Burst-DFA	92.8%	99.7%
Burst-DFA +sampling	99.0%	99.99%

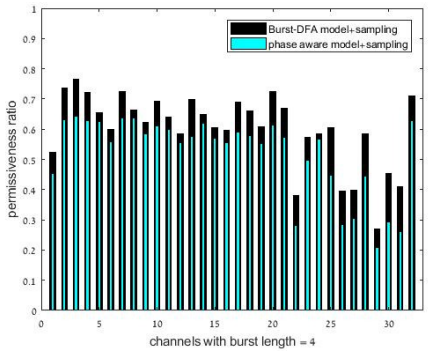

Figure 3: Permissiveness ratio R_{perm} for channels with average burst length of 4, with training set sampling

Table 1 summarizes the accuracy results of the k-phase model compared to the previous Burst-DFA model. We can learn from the table that using burst-based sampling mechanism drastically improves the accuracy of both models, and that the k-phase model is slightly less accurate than the Burst-DFA model when using sampling. If we do not use sampling, the Burst-DFA model is more accurate than the k-phase model.

As mentioned, we expect the permissiveness of the k-phase model to be lower than the permissiveness of the Burst-DFA model, since our model has more restrictions on the traffic pattern. Figure 3 shows R_{perm} for all of the channels with average burst length of 4, for the k-phase and burst-DFA models, with burst-based sampling. We can see that as expected, the level of permissiveness improves by 14% on average compared to the Burst-DFA model.

8 CONCLUSIONS AND FUTURE WORK

In our research, we analyzed a large corpus of Modbus traffic recorded at a large scale water treatment plant in the U.S. Previous research on this data corpus suggested a DFA-based model to describe the traffic; however these models did not achieve the accuracy necessary to maintain a high-fidelity system with low false alarms. In this work we showed how to improve the model fidelity to the traffic in the network while maintaining low permissiveness (detecting anomalies).

To achieve this we showed how the network traffic has different phases over time. Based on this observation, we developed a method to describe the traffic patterns at different points in time using adjacency matrices, and introduced a measure of similarity

between the traffic patterns. Then we developed a method to automatically assign the traffic into clusters based on the similarity metric introduced—i.e., an algorithm to detect the different traffic phases automatically. Next, we introduced a novel burst-based training set sampling method, which allows for training set assembly from the entire duration of the recording. The burst-based sampling method comes to ensure training on traffic from all traffic phases. We then developed a new k-phase model that incorporates the different traffic phases, by creating a unique DFA for each traffic phase detected. We also suggest a modified metric for the permissiveness of the model, that includes a method to count the unique number of walks across a set of directed graphs. The new k-phase model achieves a low False Alarm Rate, while limiting the permissiveness of the model. Finally, we showed that when using the new model, together with the burst-based sampling method, we can improve the accuracy and lower the permissiveness compared to previous models. We can achieve up to 98.9%-99.99% accuracy when using the burst-based sampling, and the k-phase model improves the permissiveness by approximately 14%.

Furthermore, the automatic identification of phase changes in the traffic has value beyond anomaly detection. Labeling these phases can help the operator understand the different states of the controlled equipment, and has the potential to bridge the semantic gap that exists in anomaly detection models.

Future work includes testing our model on other large scale datasets, testing it on longer recordings, and also testing the model's performance during true attacks on the network. We are also interested in exploring the connection between the traffic phases and the actual tasks the controlled equipment is performing and the connection between the statistical characteristics of the phase shifts and the type and designation of the controlled equipment.

ACKNOWLEDGMENTS

This work was supported by a grant from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel and the United States National Science Foundation (NSF) CNS-#1718848. This material was also supported by a grant from the Interdisciplinary Cyber-Research Center at TAU.

REFERENCES

[1] Cristina Alcaraz, Lorena Cazorla, and Gerardo Fernandez. 2015. Context-Awareness using Anomaly-based Detectors for Smart Grid Domains. In *9th International Conference on Risks and Security of Internet and Systems*, Vol. 8924. Springer International Publishing, Springer, Trento, 17–34. DOI: http://dx.doi.org/10.1007/978-3-319-17127-2_2

[2] A. Atassi, I. H. Elhajj, A. Chehab, and A. Kayssi. 2014. *The State of the Art in Intrusion Prevention and Detection, Auerbach Publications.* Auerbach Publications, Chapter 9: Intrusion Detection for SCADA Systems, 211–230.

[3] R.R.R. Barbosa, R. Sadre, and A. Pras. 2012. A first look into SCADA network traffic. In *IEEE Network Operations and Management Symposium (NOMS)*. 518–521. DOI: http://dx.doi.org/10.1109/NOMS.2012.6211945

[4] R.R.R. Barbosa, R. Sadre, and A. Pras. 2012. Towards periodicity based anomaly detection in SCADA networks. In *17th IEEE Emerging Technologies Factory Automation (ETFA)*. 1–4. DOI: http://dx.doi.org/10.1109/ETFA.2012.6489745

[5] Defense Use Case. 2016. Analysis of the cyber attack on the Ukrainian power grid. (2016).

[6] M. Caselli, E. Zambon, and F. Kargl. 2015. Sequence-aware Intrusion Detection in Industrial Control Systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. New York, NY, USA, 13–24. DOI: http://dx.doi.org/10.1145/2732198.2732200

[7] Malika Charrad, Nadia Ghazzali, Véronique Boiteau, and Azam Niknafs. 2012. NbClust Package: finding the relevant number of clusters in a dataset. *UseR! 2012* (2012).

[8] Chia-Mei Chen, Han-Wei Hsiao, Peng-Yu Yang, and Ya-Hui Ou. 2013. Defending malicious attacks in Cyber Physical Systems. In *IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA), 2013*. 13–18. DOI: http://dx.doi.org/10.1109/CPSNA.2013.6614240

[9] Noam Erez and Avishai Wool. 2015. Control Variable Classification, Modeling and Anomaly Detection in Modbus/TCP SCADA Systems. *International Journal of Critical Infrastructure Protection* 10, C (Sept. 2015), 59–70. DOI: http://dx.doi.org/10.1016/j.ijcip.2015.05.001

[10] Mustafa Faisal, Alvaro A. Cardenas, and Avishai Wool. 2016. Modeling Modbus TCP for intrusion detection. *2016 IEEE Conference on Communications and Network Security, CNS 2016* (2016), 386–390. DOI: http://dx.doi.org/10.1109/CNS.2016.7860524

[11] N. Falliere, L.O. Murchu, and E. Chien. 2011. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response* (2011).

[12] Davide Fauri, Daniel Ricardo dos Santos, Elisa Costante, Jerry den Hartog, Sandro Etalle, and Stefano Tonetta. 2017. From System Specification to Anomaly Detection (and Back). In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy (CPS '17)*. ACM, New York, NY, USA, 13–24. DOI: http://dx.doi.org/10.1145/3140241.3140250

[13] I.N. Fovino, A. Carcano, T. De Lacheze Murel, A. Trombetta, and M. Masera. 2010. Modbus/DNP3 state-based intrusion detection system. In *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*. Ieee, 729–736.

[14] Niv Goldenberg and Avishai Wool. 2013. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection* 6, 2 (2013), 63–75. DOI: http://dx.doi.org/10.1016/j.ijcip.2013.05.001

[15] D. Hadziosmanovic, D. Bolzoni, P. H. Hartel, and S. Etalle. 2011. MELISSA: Towards Automated Detection of Undesirable User Actions in Critical Infrastructures. http://eprints.eemcs.utwente.nl/20502/. In *Proceedings of the European Conference on Computer Network Defense, EC2ND 2011, Gothenburg, Sweden*. IEEE Computer Society, USA, 41–48.

[16] Amit Kleinmann and Avishai Wool. 2014. Accurate Modeling of The Siemens kleinmann2017 SCADA Protocol For Intrusion Detection And Digital Forensic. *JDFSL* 9, 2 (2014), 37–50. http://ojs.jdfsl.org/index.php/jdfsl/article/view/262

[17] Amit Kleinmann and Avishai Wool. 2017. Automatic Construction of Statechart-Based Anomaly Detection Models for Multi-Threaded Industrial Control Systems. *ACM Trans. Intell. Syst. Technol.* 8, 4, Article 55 (Feb. 2017), 21 pages. DOI: http://dx.doi.org/10.1145/3011018

[18] Ralph Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE* 9, 3 (2011), 49–51.

[19] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[20] Chen Markman, Avishai Wool, and Alvaro A. Cardenas. 2017. A New Burst-DFA Model for SCADA Anomaly Detection. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy (CPS '17)*. ACM, New York, NY, USA, 1–12. DOI: http://dx.doi.org/10.1145/3140241.3140245

[21] C. Markman, A. Wool, and A. A. Cardenas. 2018. *Temporal Phase Shifts in SCADA Networks.* Technical Report. http://arxiv.org/abs/1808.05068 arXiv:1808.05068 [cs.CR].

[22] Duc Truong Pham, Stefan S Dimov, and Cuong Du Nguyen. 2005. Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 219, 1 (2005), 103–119.

[23] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[24] R. Sommer and V. Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *IEEE Security and Privacy (SP)*. 305–316. DOI: http://dx.doi.org/10.1109/SP.2010.25

[25] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 305–316.

[26] K. A. Stouffer, J. A. Falco, and K. A. Scarfone. 2013. *Guide to Industrial Control Systems (ICS) Security.* Technical Report 800-82. National Institute of Standards and Technology (NIST), Gaithersburg, MD. DOI: http://dx.doi.org/10.6028/NIST.SP.800-82r1

[27] David I Urbina, Jairo Giraldo, Alvaro A Cardenas, Junia Valente, Mustafa Faisal, Nils Ole Tippenhauer, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. Survey and New Directions for Physics-Based Attack Detection in Control Systems. (2016).

[28] D. Yang, A. Usynin, and J.W. Hines. 2006. Anomaly-Based Intrusion Detection for SCADA Systems. In *5th Intl. Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies*. 12–16.

[29] N. Ye, Y. Zhang, and C.M. Borror. 2004. Robustness of the Markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability* 53, 1 (2004), 116–123. DOI: http://dx.doi.org/10.1109/TR.2004.823851