Understanding IEC-60870-5-104 Traffic Patterns in SCADA Networks

Chih-Yuan Lin
Dept. Comp. and Inf. Sci.
Linköping University
Linköping, Sweden
chih-yuan.lin@liu.se

Simin Nadjm-Tehrani
Dept. Comp. and Inf. Sci.
Linköping University
Linköping, Sweden
simin.nadjm-tehrani@liu.se

ABSTRACT

The IEC-60870-5-104 (IEC-104) protocol is commonly used in Supervisory Control and Data Acquisition (SCADA) networks to operate critical infrastructures, such as power stations. As the importance of SCADA security is growing, characterization and modeling of SCADA traffic for developing defense mechanisms based on the regularity of the polling mechanism used in SCADA systems has been studied, whereas the characterization of traffic caused by non-polling mechanisms, such as spontaneous events, has not been well-studied. This paper provides a first look at how the traffic flowing between SCADA components changes over time. It proposes a method built upon Probabilistic Suffix Tree (PST) to discover the underlying timing patterns of spontaneous events. In 11 out of 14 tested data sequences, we see evidence of existence of underlying patterns. Next, the prediction capability of the approach, useful for devising anomaly detection mechanisms, is studied. While some data patterns enable an 80% prediction possibility, more work is needed to tune the method for higher accuracy.

CCS CONCEPTS

• **Security and privacy** → **Network security**; • **Networks** → *Network protocols*;

KEYWORDS

SCADA; traffic patterns; IEC-60870-5-104; Probabilistic Suffix Tree (PST)

ACM Reference Format:
Chih-Yuan Lin and Simin Nadjm-Tehrani. 2018. Understanding IEC-60870-5-104 Traffic Patterns in SCADA Networks. In *CPSS'18: The 4th ACM Cyber-Physical System Security Workshop, June 4, 2018, Incheon, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3198458.3198460

1 INTRODUCTION

Modern Supervisory Control and Data Acquisition (SCADA) systems increasingly depend on information and communication technologies and become connected to the Internet to allow greater flexibility and usability. These changes make SCADA systems into attractive targets for malicious attackers [6, 7, 15].

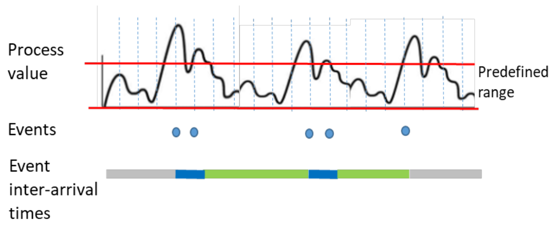

Figure 1: The repeated event inter-arrival times caused by a process value following a time-series pattern.

With the emergence of these threats, many defense mechanisms were developed to protect these critical cyber-physical systems. Most existing solutions exploit the periodic patterns that are found in synchronous communication mode between SCADA network components [5, 13, 17]. In such a communication mode, a SCADA master periodically sends requests to a field device (e.g., Remote Terminate Unit, RTU) and receives corresponding responses later. However, SCADA protocols such as IEC-104 [2] and DNP3 [1] also allow asynchronous communication mode, which means there are some spontaneous events that can be sent from a RTU without receiving any request. Lack of modeling methodologies for spontaneous events has hampered attempts to detect unusual traffic in these settings.

In order to improve the communication efficiency, most IEC-104-compatible RTUs scan monitored data in certain addresses with a fixed rate and generate spontaneous events when the monitored data has changed (e.g., from 0 to 1) or fallen outside predefined ranges. In addition to data changes caused by activation of commands, data changes can only be caused by the process subject to control. We expect that the underlying control loop for the physical process presents some repeated behaviors and generates process values containing certain time-series patterns in order to complete its regular workflow. Consequently, we speculate that the inter-arrival times of IEC-104 spontaneous events show repeated patterns when the process values contain repeated patterns as illustrated in Figure 1.

In this paper, we aim to study the inter-arrival times of IEC-104 spontaneous events using the formalism of Probabilistic Suffix Tree (PST) and analyzing the traffic regarding its phase transitions, predictability, and frequent patterns. The contributions of this paper are:

- We provide a systematic approach to model the timing of IEC-104 spontaneous traffic generated from a RTU and a process that follows the above hypothesis.
- Using data from emulated traffic in test labs, we show that there exists certain timing patterns in the IEC-104 spontaneous traffic and the patterns could provide prediction ability over a long observation time.

The rest of the paper is organized as follows. Section 2 provides the needed background about IEC-104 and PST. Section 3 discusses the related work. Section 4 describes the proposed modeling approach. Section 5 provides the overview of datasets used in this paper and presents the analysis of traffic. Finally, Section 6 concludes the paper and describes the future works.

2 BACKGROUND

This section provides an overview of IEC-104 protocol and the frame format used in this study. It also presents a brief introduction of PST with a focus of calculation of conditional and zero-order probabilities, which are used in our analysis.

2.1 IEC-60870-5-104

The IEC-104 protocol is widely used in modern SCADA systems. The basic frame in the IEC-104 protocol is called Application Protocol Data Unit (APDU) and an APDU frame can be in U, S or I format. The unnumbered control frame (U) is used for test, start or stop communication flows. The supervisory format (S) is used to perform numbered supervisory functions. The information instruction format (I) is used for sending numbered commands and information. Spontaneous events can only be sent in the I format.

Figure 2 presents the frame format for I type packets. An I format APDU is formed of the Application Protocol Control Information (APCI) and Application Service Data Unit (ASDU). The APCI contains basic information such as length of packet and sequence number and the ASDU contains the detailed attributes. There are the three attributes used for event identification and extraction in this study. *Type identification* contains the instruction code. *Cause of transmission* is always Spont for a spontaneous event. *Information object addresses* (IOA) are the addresses of the monitored data within the RTU.

2.2 Probabilistic Suffix Tree

A PST is a tree structure that can be used to learn the underlying pattern of a given sequence. Figure 3 is a PST learned from a sequence formed over a symbol set $S = \{A, B, C, D\}$ where the length of the sequence is 2000. The maximum depth of the tree is set to 2. L0 contains the root node e representing an empty string and connecting to four child nodes representing four symbols A, B, C, D in L1. At this level, each node stores the number of occurrences of this symbol in the sequence. We can easily calculate the empirical zero-order probabilities $P(A) = 387/2000, P(B) = 1304/2000, etc$ and know the probability distribution of unique symbols.

However, with the existence of patterns in a sequence, the probability of each element is conditional on the recent observed elements (i.e., the context). For the nodes in the L2 and following levels, they record the number of occurrences of a symbol σ given the context c formed of the symbols on the path in the tree up to the root node

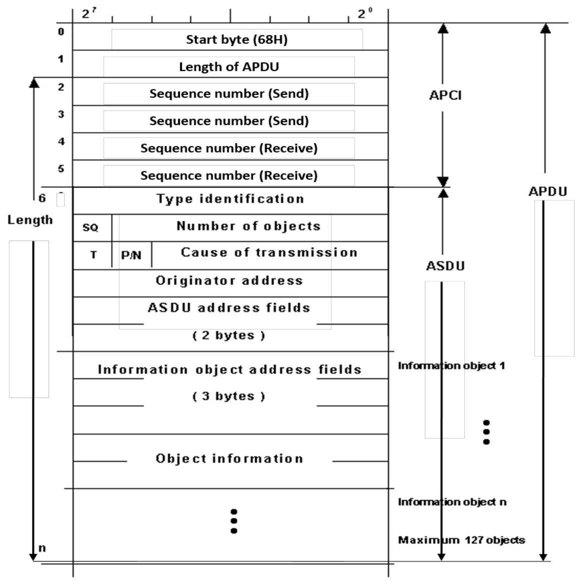

Figure 2: The I type frame format.

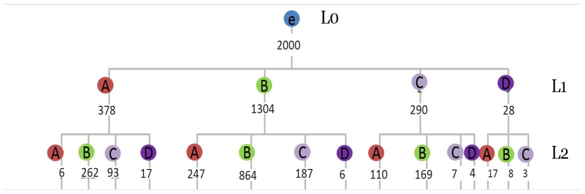

Figure 3: Example PST for a sequence of length 2000 containing 4 symbols. The maximum depth is 2.

e. This allows us to efficiently calculate the conditional probability $P(\sigma|c)$ through:

$$P(\sigma|c) \approx \frac{N(c\sigma)}{\sum_{\omega \in S} N(c\omega)} \tag{1}$$

where N(x) is the number of occurrences of a subsequence x and S denotes the symbol set as mentioned above. Thus, in the example above,

$$P(A|A) \approx \frac{6}{378} \approx 0.0159 \tag{2}$$

In our work (in section 5) we use the zero-order probabilities from an earlier observed sequence of inter-arrival times to represent the distribution of inter-arrival times and the conditional probabilities to predict the next event timing in our analysis.

3 RELATED WORK

Network analysis and characterization can be helpful for network management, creating more accurate model for simulation or traffic generation, designing and developing more efficient intrusion detection algorithms, and device fingerprinting. In the SCADA domain, Mahmood et al. [14] analyzed four traffic measurement methods regarding traffic matrix, traffic volume, traffic dynamics and traffic mixture. They proposed solutions to apply network traffic monitoring techniques to SCADA systems. This work used frequent itemset mining techniques to cluster network traffic flows. However, no analysis results of SCADA traffic was presented.

Research that contains analysis of SCADA traffic from real facilities has been published in a few instances. In 2012, Barbosa et al.[3] compared the SCADA traffic of a water facility with traditional IT traffic and found that the SCADA traffic does not exhibit characteristics used to model the traditional IT traffic including diurnal patterns, self-similarity, log-normal connection sizes, and heavy-tail distributions. In a separate work [4], Barbosa et al. compared the SCADA traffic with SNMP traffic and showed that both traffic types exhibit periodical behavior, as a consequence of the polling mechanism used to retrieve data. In 2014, Jung et al. [11] characterized the traffic of a power station network with variations in frame size, TCP connections, TCP ephemeral port number, and TCP initial sequence number. Formby et al. [8] characterized the traffic from the same environment and found TCP vulnerabilities in power grid devices. These approaches presented a high level characterization using general attributes such as TCP headers and traffic volume but not focused on a specific SCADA protocol.

There has been a particular interest in detailed characterization of SCADA traffic more recently. Goldenberg and Wool [10] used Deterministic Finite Automata (DFA) to model the cyclic behavior of Modbus. Kleinman and Wool also applied the DFA approach to S7 protocol [12]. In 2017, Formby et al. [9] characterized the power grid traffic. This work focused on DNP3 protocol and examined some common assumptions about the SCADA network such as stable traffic volume, regularity of DNP3 poll time, and long availability of SCADA devices. Our work is different from the previous work by providing analysis and detailed characterization of the traffic generated from a non-polling mechanism within a different standardized protocol. This extends our understanding of SCADA traffic characteristics.

4 PROPOSED MODELING APPROACH

Our proposed approach starts by collecting a data set from operations of a SCADA system, and uses this data set to characterize non-polling data. In addition to the data collection, it contains three main components as shown in Figure 4. First, the extractor module extracts timestamps of events with the same attributes as event sets. We will call the sequence of event inter-arrival times Δ, and denote each inter-arrival time appearing in the set by δ_i.

Second, the cluster module creates symbols (e.g., $\delta_A, \delta_B, ...$) for groups of inter-arrival times which are "similar" and uses these symbols to create symbolic sequences corresponding to Δ. We call the symbolic representation $\Delta_{categorical}$.

Finally, the symbolic sequences are input to the PST builder module and build a PST for each extracted event set.

The processes in Figure 4 where the solid rectangles are components and the round-shaped boxes are the input/output objects are described in more detail in the next subsections. The extractor is written in Python and the cluster module is in the R language. The PST builder is mainly based on method calls from the **PST**[1] package.

[1]http://CRAN.R-project.org/package=PST

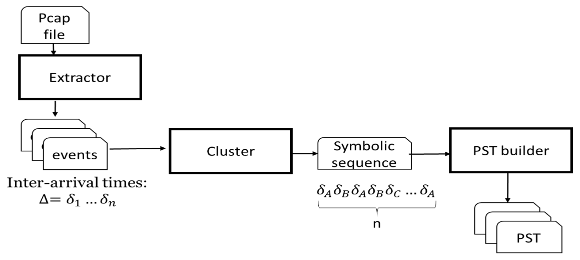
Pcap file
Extractor
events
Inter-arrival times: Δ= $\delta_1 ... \delta_n$
Cluster
Symbolic sequence
$\delta_A\delta_B\delta_A\delta_B\delta_C ... \delta_A$
n
PST builder
PST

Figure 4: Modeling flow of the system components.

4.1 Extractor

The extractor module reads a pcap file collected in a single master-RTU flow in text format and identifies spontaneous events when it finds the *Cause of transmission* of a packet is Spont as presented in section 2.1. The module then extracts the timestamps of spontaneous events using the time each packet was captured and outputs the timestamps in csv files. Each csv file represents a unique event set having the same *Type identification* (instruction code) and *Information object address* (IOA). Note that the extractor extracts events belonging to different event sets when a packet contains multiple information objects.

From the timestamps extracted the inter-arrival times (δ_i) can now be created.

4.2 Cluster

The cluster module is responsible for transforming the numeric sequence of n inter-arrival times $\Delta = \delta_1 \dots \delta_n$ in each event set into a symbolic sequence $\Delta_{categorical} = \delta_A\delta_B \dots \delta_A$ of size n formed over a symbol set S of size m. Each symbol in S is a categorized representation of a group of inter-arrival times.

The sequence Δ is divided into equal length *segments* Δ_i where the first segment will be used for learning by clustering and PST generation. The whole process contains three steps: (1) smoothing, (2) finding boundaries, and (3) sequence generation.

Smoothing. This module first uses kernel density estimation function *density()* in R standard library to smooth the distribution of Δ_1. Figure 5(a) shows part of the frequency distribution of inter-arrival times which is less than 10 seconds and Figure 5(b) is corresponding smoothing results, called $\Delta_{1\_smoothed}$. The bandwidth parameter for kernel density estimation decides the smoothness level. Its value is manually selected through a variety of tests until the space in each cluster (i.e., the distance between the right boundary and left boundary of a cluster) is almost even because the IEC-104 compatible RTUs usually scan the monitored data in a fixed rate. However, it is not set to optimize the prediction results in the later analysis section.

Finding boundaries. The next step the finds the cluster boundaries on the smoothed distribution with Algorithm 1. For the $\Delta_{1\_smoothed}$ where we can find MAX_SYMBOL_NUM or more clusters, we will only report the $MAX\_SYMBOL\_NUM - 1$ largest clusters and then categorize the others into the undefined (X) cluster. Each cluster in *ClusterList* will be denoted by a symbol in S. This limitation of the MAX_SYMBOL_NUM in Algorithm 1 gives

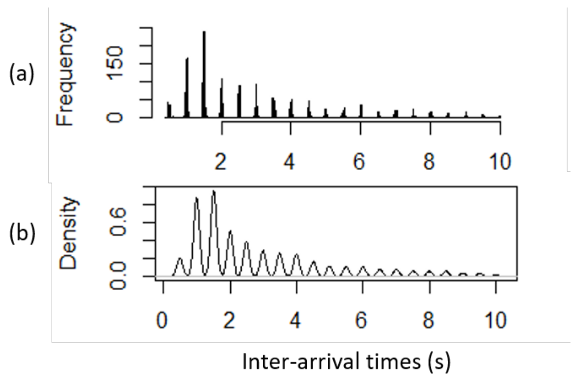

Figure 5: Distribution of inter-arrival times from a sequence within an event set in our data: (a) Histogram of $\delta_i \leq 10$ seconds. (b) The smoothed version of the sequence, bandwidth=0.008.

the number of unique symbols we can use for modeling the traffic, $m \leq MAX\_SYMBOL\_NUM$.

Algorithm 1: Finding cluster boundaries

```
Cluster;
Input  : $\Delta_{smoothed}$
Output : A list of cluster boundaries
$ClusterList \leftarrow empty$ // list for output
for $i := 1$ to $MAX\_SYMBOL\_NUM - 1$ do
    $peak \leftarrow IndexOfMaxElement(\Delta_{smoothed})$;
    $L = R = peak$;
    while $\Delta_{smoothed}[R+1] < \Delta_{smoothed}[R]$ do
        $R \leftarrow R + 1$;
    end
    while $\Delta_{smoothed}[L-1] < \Delta_{smoothed}[L]$ do
        $L \leftarrow L - 1$;
    end
    $ClusterList[i] \leftarrow (L, R)$
    if $(L == R)$ then
        break;
    end
end
```

Sequence generation. Finally, the next step categorizes each inter-arrival time δ_i in Δ by mapping it into a symbol in S and generates sequence $\Delta_{categorical}$.

4.3 PST builder

The PST builder module uses the *pstree()* function of **PST** package to learn the PST models from the output $\Delta_{1\_categorical}$ without setting any pruning or smoothing parameters. The height of the PST has to be fixed in order to manage the computational complexity. The datasets typically include repeated patterns of few symbols long which may guide the choice of height to capture those frequent patterns. This has to be determined experimentally.

5 ANALYSIS

This section first presents the overview of the used datasets and then describes the analysis part of the work in detail. Our analysis consists of three different aspects that provide a detailed characterization of the datasets. The phase transition analysis is used to show that there exists a few numbers of phases, in which the distribution of inter-arrival times are relatively stable (Section 5.2). The predictability analysis validates the existence of sequence patterns of inter-arrival times by comparing the prediction capability of the built PST models and pseudo models. A pseudo model is built upon a dataset that is synthetically generated from the zero-order probabilities of the built model using the random walk methodology. Therefore, the generated dataset follows the same distribution as the original dataset but there's no dependencies between any pair of adjacent symbols (Section 5.3). The third analysis, the frequent pattern analysis, presents the most frequent patterns for different event sets and explains the predictability analysis results (Section 5.4).

5.1 Datasets and parameter settings

In this study we analyze two different datasets: One is from a small-scale SCADA laboratory maintained by the Department of Industrial Information and Control Systems at KTH (Royal Institute of Technology) with real hardware components [16]. This laboratory contains 4 RTUs but the available dataset includes only traces of RTU 1 and 4, which are further used for modeling and analysis. The second dataset is from the virtual SCADA network RICS-EL that is developed in our project. RICS-EL emulates an electricity utility network extending FOI[2] Cyber Range And Training Environment (CRATE). The process dynamics are generated and provided by a major SCADA vendor in an emulated environment that uses their product. Both of the datasets are network traces in the pcap format.

Table 1 shows the overview of the used traces and the extracted event sets. We separate the extracted events into roughly two-hour segments with the following equation:

$$\text{number of segments} = \lfloor duration/2 \rfloor, \tag{3}$$

where the duration is the length of time (hours) over which the event sequence was collected. We then use the first segment for training the model and the remaining ones for analysis. Since the learning sequence may be formed of underlying patterns with missed or additional elements, the size of the training segment must be large enough to avoid biased learning results. Therefore, we only use event sets where the number of elements in a two-hour segment is larger than 100 events. We give each used event set a unique name and refer to it with its name in the rest of the paper.

In our experiments we chose the height of the PSTs to be 6 since we found repeated patterns found of 2-4 symbols long in our datasets. The **PST** library we used has a default value of 12 symbols as the limit in Algorithm 1. The default value was kept to explore its suitability for the model and retain the known performance properties of the package.

[2]Swedish Defense Research Agency (https://www.foi.se)

Table 1: Overview of used traces and the extracted event sets

Traces	Duration	Size	Instruction	IOA	# of events	Name
KTH-RTU1	6 days	468,199KB	M_ME_NA_1	1	387340	K_1_1
				2	1130288	K_1_2
				3	467223	K_1_3
				4	1132359	K_1_4
KTH-RTU4	6 days	278,519KB	M_ME_NA_1	2	333676	K_4_2
				3	1141439	K_4_3
				4	177922	K_4_4
RICS	12 days	448,002KB	M_ME_NA_1	10001	2	—
				10002	18899	R_02
				10004	2	—
				10005	39512	R_05
				10010	1057	—
				10011	236315	R_11
				10012	7059	—
				10013	1233	—
				10014	17256	R_14
				10015	3835	—
				10016	17609	R_16
				10017	3818	—
				10020	1	—
				10091	198844	R_91
				10092	207893	R_92

5.2 Phase transitions

As observed in earlier works [10, 12], SCADA traffic sometimes contains phase transitions. In this study, we define a phase as a period of time that the distribution of inter-arrival times is stable. A phase transition happens when the changes of the distribution falls outside a certain range.

Because the transition probabilities from a L0 node to the L1 nodes represents the distribution of the unique symbols (i.e., inter-arrival times) in a PST as described in section 2.2, we build a PST for each $\Delta_{i\_categorical}$ and calculate the distances from the L0 node of the first segment to every L0 node of the remaining segments using the following steps. For any given two L0 nodes e_1, e_2 from two different trees, there exists two sets of L1 symbols $S^1 = \{\sigma_1^1, \ldots, \sigma_p^1\}$ and $S^2 = \{\sigma_1^2, \ldots, \sigma_q^2\}$. For each L1 symbol, the transition probability is denoted as $p(\sigma_i^k)$ for $k \in \{1, 2\}$. The L0 distance D between e_1, e_2 is defined as:

$$D = \sum_{\sigma_i^1 = \sigma_i^2} (p(\sigma_i^1) - p(\sigma_i^2))^2 + \sum_{\sigma_i^1 \neq \sigma_i^2} p(\sigma_i^k)^2 \tag{4}$$

The left term of equation (4) captures the difference between frequency of the same symbol appearing at L1 level in two different segments. The right term captures the case where some symbol appears in one tree but not in the other. Hence, the larger the distance the more different the patterns are in the two segments.

Figure 6 shows the changes of L0 distance over time for different event sets from Table 1. We observed five groups of traffic pattern. The group I event sets contain strongly cyclic patterns of change in the distribution of inter-arrival times. The group II event sets contain weakly cyclic patterns in the sense that the changes of the distribution are relatively small. The group III event sets have almost no change of distribution. The group IV presents irregular bursts over time. Finally, the group V event sets contain a series of bursts for a period of time, which are collectively considered as a phase. The red lines indicate the starting or stopping point of another phase.

In order to discover phase transitions, for the group V event sets, we find the starting and stopping point of another phase by detecting bursts when the changes of L0 distance exceeds a phase transition threshold (PTT). We do this by locating the first and last burst, and setting the interval between the first and last burst as another phase if the last burst is not found in the end of the event set.

In order to standardize the threshold selection procedure, we consider an acceptable deviation ϵ which is small enough so that even if it accumulates for every symbol it will not signify a phase transition. This value has to be chosen experimentally and used to determine phase transition threshold through equation (5).

$$PTT = m * \epsilon^2 \tag{5}$$

where m is the previously mentioned number of unique symbols learned in the first segment (Section 4.2).

Next section presents the predictability of event sets in different groups, together with the selected ϵ and length of phases for group V. For the sake of clarity, we refer to the first phase of an event set as ϕ for the following analysis.

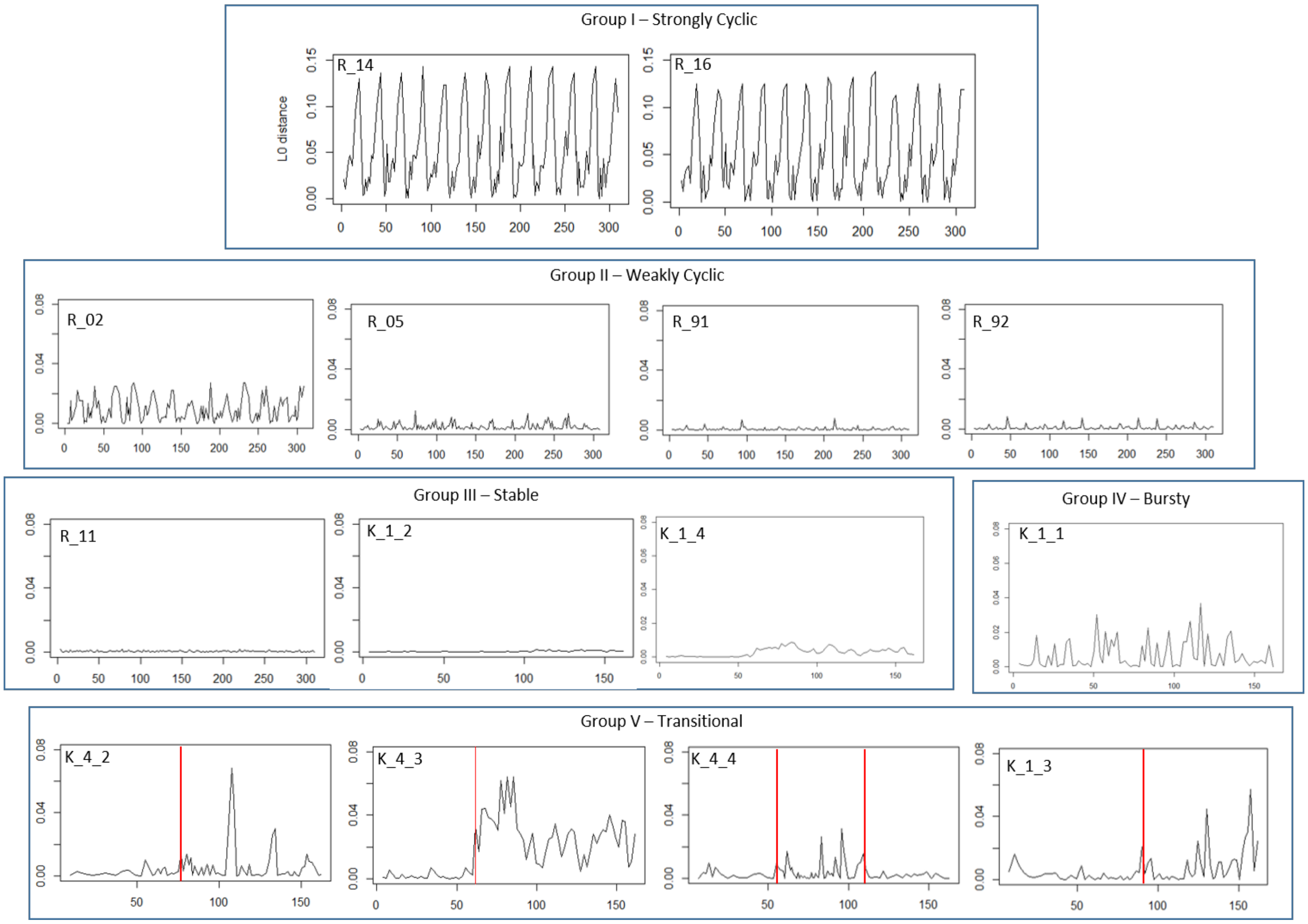

Figure 6: L0 distances over time (hours) for different event sets.

5.3 Predictability

There are two metrics used in the predictability analysis, prediction accuracy and Kappa value. To calculate the prediction accuracy of each segment, we run a sliding window of size 6 (tree height) within each $\Delta_{i\_categorical}, i \neq 1$, and input the sequence in the sliding window as a context to the *predict()* function. The function queries the built PST from $\Delta_{1\_categorical}$ and returns the probability of each symbol calculated by equation (1). We choose the symbol with the highest likelihood as our predicted value for a position. In each segment, we calculate the observed prediction accuracy with equation (6) using the resulting confusion matrix. Here, N is the number of predictions performed based on the size of the sliding window and the size of the segment. c is the number of unique symbols appearing in predictions or actually in the segment, which is the number of rows and columns in the matrix. n_{ij} is the number of times the symbol i (ground truth) is predicted as j.

$$\text{observed accuracy} = \frac{\sum_{i=1}^{c} n_{ii}}{N} \tag{6}$$

In order to get the Kappa value, we also calculate the expected accuracy by summing up the probability that the predicted value and the ground truth happens to be the same.

$$\text{expected accuracy} = \sum_{i=1}^{c} \left(\frac{n_{i+}}{N} \times \frac{n_{+i}}{N}\right) \tag{7}$$

where n_{i+} is the total number of times the symbol i appears in the segment and n_{+i} is the total number of times any symbol is predicted as i. Kappa value is

$$Kappa = \frac{\text{observed accuracy} - \text{expected accuracy}}{1 - \text{expected accuracy}} \tag{8}$$

The Kappa value of the prediction results is used to indicate when the observed accuracies are higher than the expected accuracies (Kappa is not around 0).

The observed prediction accuracy and Kappa value are applied to both the built model and the pseudo mode (the randomly generated model described at the beginning of section 5) for each event set in order to confirm the existence of sequence patterns. For a pseudo model we expect the Kappa values to be around 0 and for a built model we expect the Kappa values and prediction accuracies to be higher than the values produced by the pseudo model due to the presence of repeated sequences.

Figure 7 presents the results of the predictability analysis over time (hours). The Kappa value of a pseudo model is always around 0 for all the event sets as expected. Except for the group I event sets and R_05, all of the event sets present ideal results: the prediction accuracies and Kappa values of the built model are higher than the values of the pseudo model in the whole duration even for the group V event sets with phase transitions.

Table 2 summarizes the selected ϵ, length of ϕ, and the prediction accuracies for the four event sets with phase transitions. We can see that K_4_3 and K_4_4 have better average accuracy and Kappa value in the first phase (column 4 and 5) than the whole duration. However, the impact of phase transitions of K_4_2 and K_1_3 is not significant. This may be caused by the low prediction capability of the built models. That is, the built models only capture a small amount of underlying patterns and these patterns are not significantly changed after the phase transitions.

On the other hand, the group I event sets and R_05 do not present such ideal behaviors. In the case of R_05 the Kappa values and accuracies of built model are better than the values of the pseudo model for most of the time, but in the case of the R_14 and R_16, the values of the built model are no better than the values of the pseudo model. All of them have cyclic drops of prediction ability. We present a further study and explanations of these cases in the next section.

5.4 Frequent patterns

In this section, we look into the frequent patterns of the built models for all event sets by using the *pmine()* function of **PST** package in order to understand the challenges of modeling spontaneous events and the limitations of our approach. For cases where the predictability was high we expect a frequently appearing set of patterns.

Figure 8 illustrates pattern mining results. For each dataset we have depicted the 10 most frequent patterns of length 6 (corresponding to the tree height and sliding window in the previous sections) extracted by mining, shown as a block of colored cells. The y axis of a block gives the order of the frequent patterns. 1 denotes the most frequent pattern. The symbol 0-10 in the legend denotes the inter-arrival times located in the first to eleventh largest cluster. Symbol 11 denotes the inter-arrival times in the undefined cluster X. Note that the same symbol can represent different time for different event sets. We can conclude the following points from Figure 8.

- For the event sets that present stable L0 distance distributions in the whole duration or before phase transition, but have low average prediction accuracy around 20% (K_1_2, K_1_4, K_4_2), we can observe a lot of symbols from the undefined cluster in the learned frequent patterns (shown as dark yellow cells). This indicates the need of a larger symbol set, which is currently limited by the default setting of the used **PST** package.
- In contrast, the event sets that show non-ideal results in the predictability analysis (R_14, R_16, R_05) have just a few or no symbols from the undefined cluster in the learned frequent patterns. This was somewhat unexpected. However, after examining the whole event sets, we found that the number of undefined symbols highly increases in the non-learning segments. This indicates the need for longer learning period up to the length of the cycle and perhaps leads to the need of larger symbols sets as well. Since the length of the observed cycles is around 24 hours, the needed size of sequence for prediction could be over 3900 events.
- There are 6 event sets, R_91, R_92, R_11, K_1_1, K_4_3 and K_4_4, that show moderate to high level of prediction accuracy (40%-80%). They have two common characteristics. First, there is no undefined symbol in the learned frequent patterns of these event sets. This means the undefined symbols occur infrequently. Second, there are some short underlying patterns that are repeatedly visible. For example in R_91 and R_92, we can find the unique and most frequent sequence "0002" appearing several times. For K_4_4, there are instances of "10" and "12" sequences appearing repeatedly.

5.5 Insights

This section briefly summarizes the insights obtained after our analysis.

Before performing this work we were not aware of any inter-arrival patterns for spontaneous events in our datasets. Our phase transition analysis disclosed 5 categories (groups I to V from Figure 6) in our datasets. Getting to know your data is the first step in building suitable anomaly detection systems.

Our hypothesis was that predictability is a function of length of learning interval and potential patterns in data, but we had no idea about any patterns in our data. Having done the frequent pattern analysis (Figure 8) we discovered the diverse variations of patterns in our data. These are orthogonal to the category groups. Our learning period, using 2-hour segments, turned out to be too short for cyclic and some weakly cyclic datasets. The analysis led us to a more detailed examination of some datasets where 24 hours (corresponding to the 3900 events above) would be a more suitable learning interval.

Based on this work PST can be a useful method for getting insights on timing patterns in collected data. In 11 out of 14 of our event sets we could get reasonable predictability. Hence, it is worthwhile to explore anomaly detection of spontaneous events at least in these cases.

6 CONCLUSIONS AND FUTURE WORK

This paper applies probabilistic suffix tree to model and analyze two emulated IEC-104 datasets regarding the timing of the spontaneous events.

Using the changes of distribution of different event inter-arrival times over time, we categorize the traffic into 5 different groups, namely strongly cyclic, weakly cyclic, stable, bursty, and phase transitional groups. Using the prediction capability of the PSTs we tried to ascertain whether patterns present in the learned data set segments could be seen to appear in the test segments. 11 out of 14 event sets show evidence of presence of underlying patterns.

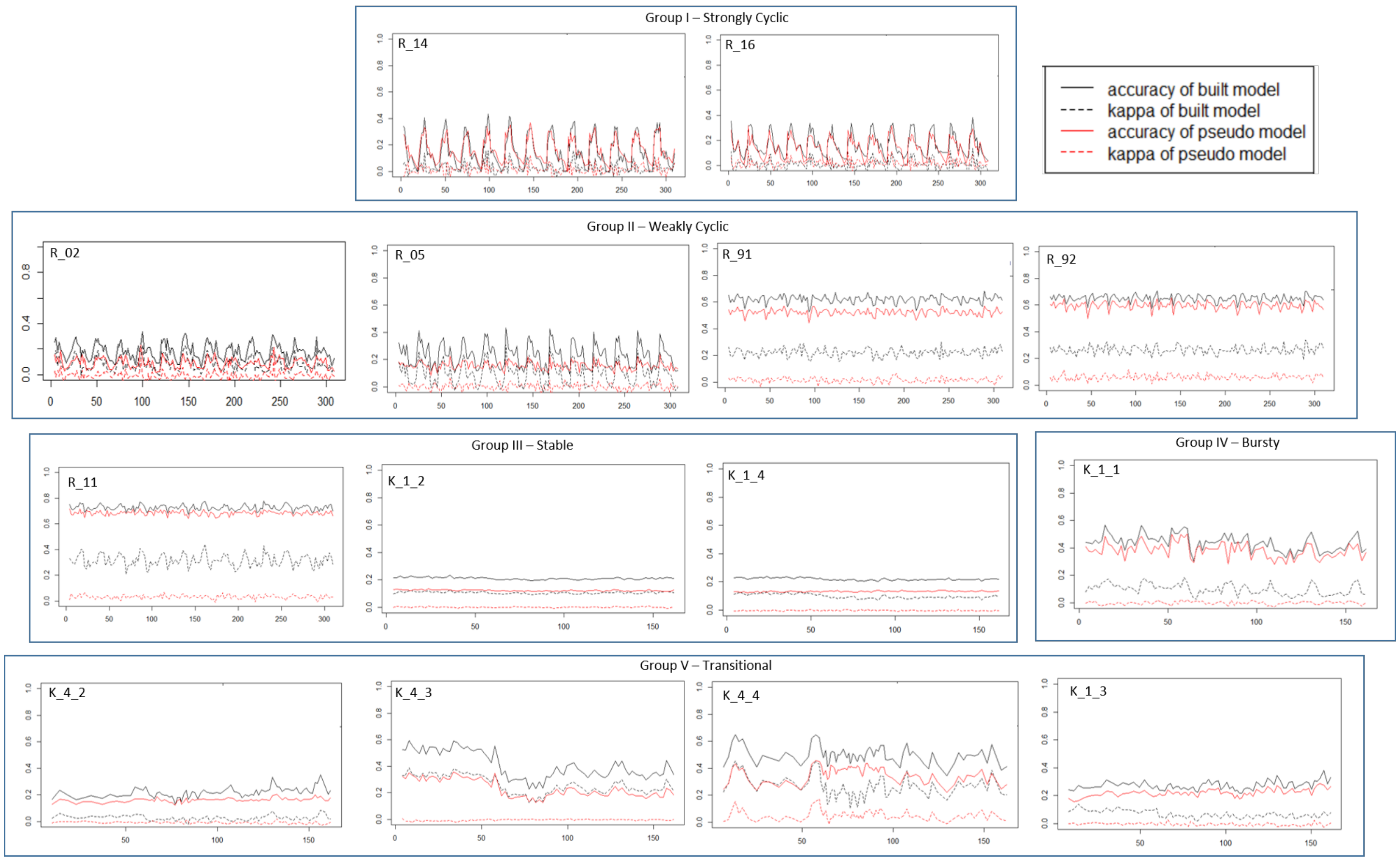

Figure 7: Predictability analysis results.

Table 2: Summary of predictabilities in the first phase and the whole event sets

Event set	ϵ	Len. of ϕ	Avg. accuracy in ϕ [%]	Avg. Kappa in ϕ	Avg. accuracy (full) [%]	Avg. Kappa (full)
K_4_2	0.03	77 hours	20	0.03	22	0.02
K_4_3	0.03	61.7 hours	52	0.33	41	0.26
K_4_4	0.03	61.6 hours	52	0.33	50	0.26
K_1_3	0.04	89.9 hours	27	0.07	27	0.06

6 event sets show moderate to high level of prediction accuracy (40%-80%) with our modeling approach.

Although the datasets used in this paper are emulated, they show a diverse set of possible timing patterns in SCADA networks. Irrespective of dataset, the probability distributions of event inter-arrival times present clear peaks similar to those presented in Figure 5. These distributions can be stable over tens to hundreds of hours. Based on the observation of the positive predictability results we have an indication that building an anomaly detector to detect inserted attack packets from machine-generated streams is worth pursuing in more detail.

While the preliminary results show the idea of modeling timing patterns of spontaneous events using PST is satisfactory, there are some challenges to model the timing of spontaneous events as well as some observed limitations in our approach. PSTs have a computation complexity which may not work with tree heights exceeding a certain level. Considering the computation capability of a current computer, creating and using an accurate timing model of spontaneous events with a long cycle (e.g., 3900 events in a 24-hour cyclic pattern observed in one of our event sets) becomes quite challenging. The current setting in the *ptree()* function of **PST** package limits the maximum number of used symbols to 12. This means we can only model the inter-arrival times with the eleven highest probabilities. This setting works when the distribution of inter-arrival times are centered. A dispersed distribution of inter-arrival times will cause too many undefined symbols and lose the prediction ability. In the current state, the approach was not applicable to all variation of event sets in our data.

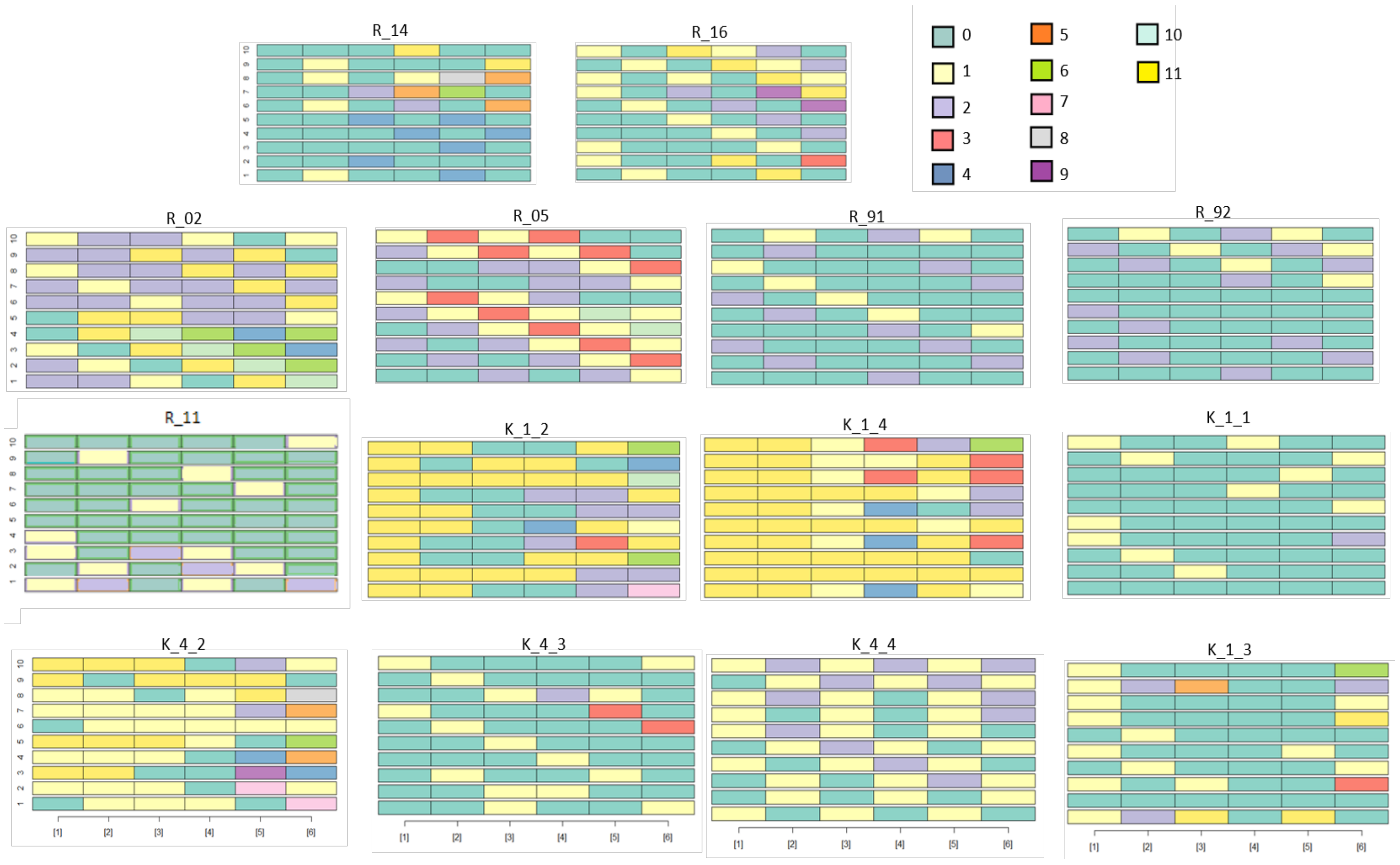

Figure 8: Pattern mining results.

This work discovered the potential of modeling the timing of IEC-104 spontaneous events. The obvious future work includes studying the properties of a PST library with a higher maximum number of symbols limitation. This allows to explore the strongly cyclic group of traffic as a foundation for future research on anomaly detection.

We plan to collect longer datasets from real SCADA systems in different domains, conduct an empirical analysis, and compare the results with the insights from the synthetic datasets. This will be a basis for generating emulated data in our test bed that closely resembles the real datasets. Understanding different patterns allows creating data that resembles data from different domains, stakeholders, or SCADA configurations. This means specially crafted attacks can be generated and studied in a "safe" environment that resembles real networks, and mitigation strategies created to deal with them.

ACKNOWLEDGMENTS

This work was carried out in the Centre for research on Resilient Information and Control Systems (RICS) in Sweden (www.rics.se) and was supported by the Swedish Civil Contingencies Agency (MSB). The authors wish to extend their gratitude to the members of the NCS3 team at Swedish Defence Research Agency (FOI) for collaborating on creation of our testbed, and acknowledge the data collection in the KTH testbed in a previous collaboration.

REFERENCES

[1] [n. d.]. Distributed Network Protocol 3.0. ([n. d.]). https://www.dnp.org/pages/aboutdefault.aspx
[2] [n. d.]. IEC 60870-5-104. ([n. d.]). https://webstore.iec.ch/publication/25035
[3] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. 2012. Difficulties in Modeling SCADA Traffic: A Comparative Analysis. In *Passive and Active Measurement. PAM 2012. Lecture Notes in Computer Science*, Vol. 7192. Springer.
[4] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. 2012. A first look into SCADA network traffic. In *IEEE Network Operations and Management Symposium (NOMS)*.
[5] Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras. 2016. Exploiting Traffic Periodicity in Industrial Control Networks. *Int. J. Crit. Infrastruct. Prot.* 13, C (June 2016), 52–62. https://doi.org/10.1016/j.ijcip.2016.02.004
[6] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. 2011. *Duqu: A Stuxnet-like malware found in the wild.* Technical Report. Laboratory of Cryptography and System Security (CrySyS Lab), Budapest University of Technology and Economics Department of Telecommunications.
[7] Nicolas Falliere, Liam O Murchu, and Eric Chien. 2011. *W32.Stuxnet Dossier.* Technical Report. Symantec, Mountain View.
[8] David Formby, Sang Shin Jung, John Copeland, and Raheem Beyah. 2014. An Empirical Study of TCP Vulnerabilities in Critical Power System Devices. In *SEGS '14 Proceedings of the 2nd Workshop on Smart Energy Grid Security.* 39–44.
[9] David Formby, Anwar Walid, and Rahhem Beyah. 2017. A Case Study in Power Substation Network Dynamics. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 19 (2017).
[10] 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. https://doi.org/10.1016/j.ijcip.2013.05.001

[11] Sang Shin Jung, David Formby, Carson Day, and Raheem Beyah. 2015. A first look at machine-to-machine power grid network traffic. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on.*

[12] Amit Kleinmann and Avishai Wool. 2014. Accurate Modeling of the Siemens S7 SCADA Protocol for Intrusion Detection and Digital Forensic. *The Journal of Digital Forensics, Security and Law* 9, 2 (2014).

[13] Amit Kleinmann and Avishai Wool. 2016. A Statechart-Based Anomaly Detection Model for Multi-Threaded SCADA Systems.. In *International Conference on Critical Information Infrastructures Security (CRITIS 2015).*

[14] Abdun Naser Mahmood, Christopher Leckie, Jiankun Hu, Zahir Tari, and Mohammed Atiquzzaman. 2010. Network Traffic Analysis and SCADA Security. *Handbook of Information and Communication Security* (2010), 383–405.

[15] sKyWIper Analysis Team. 2012. *sKyWIper (a.k.a. Flame a.k.a. Flamer): A complex malware for targeted attacks.* Technical Report. Laboratory of Cryptography and System Security (CrySyS Lab), Budapest University of Technology and Economics Department of Telecommunications.

[16] Robert Udd, Mikael Asplund, Simin Nadjm-Tehrani, Mehrdad Kazemtabrizi, and Mathias Ekstedt. 2016. Exploiting Bro for Intrusion Detection in a SCADA System.. In *CPSS '16 Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security.*

[17] Alfonso Valdes and Steven Cheung. 2009. Communication pattern anomaly detection in process control systems. In *IEEE Conference on Technologies for Homeland Security. (HST '09).*