# Automatic Construction of Statechart-Based Anomaly Detection Models for Multi-Threaded SCADA via Spectral Analysis

Amit Kleinmann
Tel Aviv University
Ramat Aviv, Tel-Aviv, Israel
ak7jar@gmail.com

Avishai Wool
Tel Aviv University
Ramat Aviv, Tel-Aviv, Israel
yash@acm.org

## ABSTRACT

Traffic of Industrial Control System (ICS) between the Human Machine Interface (HMI) and the Programmable Logic Controller (PLC) is highly periodic. However, it is sometimes multiplexed, due to multi-threaded scheduling.

In previous work we introduced a *Statechart* model which includes multiple Deterministic Finite Automata (DFA), one per cyclic pattern. We demonstrated that Statechart-based anomaly detection is highly effective on multiplexed cyclic traffic when the individual cyclic patterns are known. The challenge is to construct the Statechart, by unsupervised learning, from a captured trace of the multiplexed traffic, especially when the same symbols (ICS messages) can appear in multiple cycles, or multiple times in a cycle. Previously we suggested a combinatorial approach for the Statechart construction, based on Euler cycles in the Discrete Time Markov Chain (DTMC) graph of the trace. This combinatorial approach worked well in simple scenarios, but produced a false-alarm rate that was excessive on more complex multiplexed traffic.

In this paper we suggest a new Statechart construction method, based on spectral analysis. We use the Fourier transform to identify the dominant periods in the trace. Our algorithm then associates a set of symbols with each dominant period, identifies the order of the symbols within each period, and creates the cyclic DFAs and the Statechart.

We evaluated our solution on long traces from two production ICS: one using the Siemens S7-0x72 protocol and the other using Modbus. We also stress-tested our algorithms on a collection of synthetically-generated traces that simulate multiplexed ICS traces with varying levels of symbol uniqueness and time overlap. The resulting Statecharts model the traces with an overall median false-alarm rate as low as 0.16% on the synthetic datasets, and with zero false-alarms on production S7-0x72 traffic. Moreover, the spectral analysis Statecharts consistently out-performed the previous combinatorial Statecharts, exhibiting significantly lower false alarm rates and more compact model sizes.



*CPS-SPC'16, October 28 2016, Vienna, Austria*

 ISBN 978-1-4503-4568-2/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2994487.2994490

## Keywords

ICS; SCADA; Network-intrusion-detection-system; Statechart; Siemens; S7; Modbus

## 1. INTRODUCTION

### 1.1 Background

Industrial Control Systems (ICS) are used for monitoring and controlling numerous industrial systems and processes. In particular, ICS are used in critical infrastructure assets such as chemical plants, electric power generation, transmission and distribution systems, water distribution networks, and waste water treatment facilities. ICS have a strategic significance due to the potentially serious consequences of a fault or malfunction.

ICS typically incorporate sensors and actuators that are controlled by Programmable Logic Controllers (PLCs), and which are themselves managed by a Human Machine Interface (HMI). PLCs are computer-based devices that were originally designed to perform the logic functions executed by electromechanical hardware (relays, switches, mechanical timer, and mechanical counters). PLCs have evolved into controllers with the capability of controlling the complex processes used for discrete control in discrete manufacturing. The NIST Guide to ICS Security [30] explains that ICS is a general term that encompasses several types of control systems, including Programmable Logic Controllers (PLC), Distributed Control Systems (DCS), Supervisory Control And Data Acquisition (SCADA) systems, and other control system configurations. An automation system within a campus is usually referred to as a DCS, while SCADA systems typically comprise of different stations distributed over large geographical areas.

ICS were originally designed for serial communications, and were built on the premise that all the operating entities would be legitimate, properly installed, perform the intended logic and follow the protocol. Thus, many ICSes have almost no measures for defending against deliberate attacks. Specifically, ICS network components do not verify the identity and permissions of other components with which they interact (i.e., no authentication and authorization mechanisms); they do not verify message content and legitimacy (i.e., no data integrity checks); and all the data sent over the network is in plaintext (i.e., no encryption to preserve confidentiality). Therefore, deploying an anomaly detection system in an ICS network is an important defensive measure.

## 1.2 Related work

### 1.2.1 Attacks

The susceptibility of ICS systems to attacks has been known for many years. [7] describe different attack trees on SCADA based on the Modbus/TCP protocol. They found that compromising the slave (PLC) or the master (HMI) has the most severe potential impact on the SCADA. For instance, an attacker that gains access to the SCADA could identify as the HMI and change data values in the PLC. Alternately, an attacker can perform a Man In The Middle attack between a PLC and HMI and "feed" the HMI with misleading data, allegedly coming from the exploited PLC.

Digital attacks that cause physical destruction of equipment do occur. Perhaps most notably is the attack on an Iranian nuclear facility in 2010 (Stuxnet) to sabotage centrifuges at a uranium enrichment plant. The Stuxnet malware [13, 22] implemented a water–hammer style attack by changing centrifuge operating parameters in a pattern that damaged the equipment – while sending normal status messages to the HMI to hide the fact that an attack is under way. [23] describes a more recent event, where hackers had struck an unnamed steel mill in Germany, by manipulating and disrupting control systems to such a degree that a blast furnace could not be properly shut down, resulting in "massive"-though unspecified-damage.

At BlackHat USA 2015 Klick et al. [21] demonstrated injection of malware into a SIMATIC S7-300 PLC without service disruption. In a follow on work, [29] demonstrated the feasibility of a PLC worm. The worm spreads internally from one PLC to other target PLCs. During the infection phase the worm scans the network for new targets (PLCs).

### 1.2.2 Anomaly Detection

A survey of techniques related to learning and detection of anomalies in critical control systems can be found in [2].

While most of the current commercial network intrusion detection systems (NIDS) are signature-based, i.e., they recognize an attack when it matches a previously defined signature, anomaly-based NIDS "are based on the belief that an intruder's behavior will be noticeably different from that of a legitimate user" [25].

Different kinds of Anomaly Intrusion Detection models have been suggested for SCADA systems. [34] used an Auto Associative Kernel Regression (AAKR) model coupled with the Statistical Probability Ratio Test (SPRT) and applied them on a SCADA system looking for matching patterns. The model used numerous indicators representing network traffic and hardware-operating statistics to predict the 'normal' behavior.

Several recent studies [3, 9] suggest anomaly-based detection for SCADA systems which are based on Markov chains. However, [35] showed that although the detection accuracy of this technique is high, the number of False Positive values is also high, as it is sensitive to noise. [16] used the logs generated by the control application running on the HMI to detect anomalous patterns of user actions on process control application.

[14] have presented a state-based intrusion detection system for SCADA systems. Their approach uses detailed knowledge of the industrial process' control to generate a system virtual image. The virtual image represents the PLCs of a monitored system, with all their memory registers, coils, inputs and outputs. The virtual image is updated using a periodic active synchronization procedure and via a feed generated by the intrusion detection system (i.e., known intrusion signatures).

Model-based anomaly detection for SCADA systems, and specifically for Modbus traffic, was introduced by [10]. They designed a multi-algorithm intrusion detection appliance for Modbus/TCP with pattern anomaly recognition, Bayesian analysis of TCP headers and stateful protocol monitoring, complemented with customized Snort rules [27]. In subsequent work, [31] incorporated adaptive statistical learning methods into the system to detect for communication patterns among hosts and traffic patterns in individual flows. Later [6] integrated these intrusion detection technologies into the EMERALD event correlation framework [26].

[28] discuss the surprising imbalance between the extensive amount of research on machine learning-based anomaly detection versus the lack of operational deployments of such systems. One of the reasons for that, by the authors, is that the machine learning anomaly detection systems are lacking the ability to bypass the "semantic gap": The system "understands" that an abnormal activity has occurred, but it cannot produce a message that will elaborate, helping the operator differentiate between an abnormal activity and an attack.

[12] developed an anomaly detection system that detects irregular changes in SCADA control registers' values. The system is based on an automatic classifier that identifies several classes of PLC registers (Sensor registers, Counter registers and Constant registers). Parameterized behavior models were built for each class. In its learning phase, the system instantiates the model for each register. During the enforcement phase the system detects deviations from the model.

### 1.2.3 Periodicity

Barbosa et al. [4] analyzed SCADA traces they collected at two different water treatment and distribution facilities. To check for periodicity, they carried out a Fourier analysis for the packet time series for each source IP address and the aggregate of all sources.The SCADA datasets exhibited periodicity with dominant period components at 50s, and 60s. They concluded that SCADA traffic presents remarkably regular time series, due to the fact that the majority of the traffic sources generate data in a periodical fashion.

In a follow-on work, Barbosa et al. [5] provided a proof of concept of their periodicity learning. Through manual inspection they selected the high energy frequencies for the anomaly detection phase, and discarded the others. However their analysis considered only the transport level of the traffic without looking at the SCADA protocol semantics. Furthermore, they did not consider an essential characteristic of the SCADA traffic - the proper message order of the periodic packet-sequence. Kleinmann et al.[19] explained that beyond request-response cyclic patterns, periodic traffic can also be formed in a different way. Certain SCADA protocols such as the Siemens S7 allow the clients to "subscribe" to a certain register range at the server, after which the server asynchronously sends a stream of notifications with the values of the subscribed registers.

### 1.2.4 Automata-based models

Goldenberg & Wool [15] developed a model-based ap-

proach (the GW model) for Network anomaly detection based on the normal traffic pattern in Modbus SCADA networks using a Deterministic Finite Automata (DFA) to represent the cyclic traffic. Following this approach the SCADA messages are modeled both in isolation and also by their sequence order. Subsequently, in [18] we demonstrated that a similar methodology is successful also in SCADA systems running the Siemens S7 protocol.

[8] proposed a methodology to model sequences of SCADA protocol messages as Discrete Time Markov Chains (DTMCs). They built a state machine whose states model possible messages, and whose transitions model a "followed-by" relation. 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 [4, 15, 18] of a substantial sequentiality in the SCADA communications. However, unlike [15, 18] they did not observe clear cyclic message patterns. The authors hypothesized that the difficulties in finding clear sequences is due to the presence of several threads in the HMI's operating system that multiplex requests on the same TCP stream.

Modeling the network traffic patterns of multiplexed SCADA streams, as observed by [8], using DFA for anomaly detection typically produces a very large DFA, and a high false-alarm rate. Kleinmann et al. [19] introduced a modeling approach for such SCADA streams, using *Statechart DFAs*: the *Statechart* includes multiple DFAs, one per cyclic pattern. Each DFA is built using the learning stage of the GW model. Following this model, incoming traffic is de-multiplexed into sub-channels and sent to the respective DFAs. We showed that if the correct DFAs are known the *Statechart* model drastically reduced both the false-alarm rate and the learned model size in comparison with the naive single-DFA model. However, the question of how to automatically learn multiplexed traffic and construct the *Statechart* remained an open question.

Our first attempt to automatically construct the *Statechart* was presented in [20]. We used unsupervised learning algorithm that builds a DTMC from the stream. It then splits the symbols into sets, one per multiplexed cycle, based on symbol frequencies and node degrees in the DTMC graph. Next it extracts Euler cycles for each cyclic pattern in a PLC-HMI channel. The result patterns are then merged to create the *Statechart* DFAs (one DFA per Euler cycle). This combinatorial approach worked reasonably well on synthetically–generated traces, however it still suffered from a too–high false alarm rate, and it was not robust enough to handle real S7-0x72 traffic (see Section 6.2) and complex scenarios where some of the patterns overlap, i.e., the same symbols appear in different cyclic patterns. The major reason for the false alarms in these scenarios was the inaccuracy of channel splitting to sub-channel (pattern) components caused by the strict combinatorial requirements that characterize Euler cycles.

## 1.3 Contributions

In this paper, we suggest a novel Statechart construction algorithm, based on spectral analysis that provides near-optimal performance, which is always superior to that of the combinatorial approach.

At our starting point we convert the captured trace into a "signal", and apply the Fourier transform to it to identify the dominant periods in the frequency domain. We identify the cyclic pattern by switching back to the time domain, determining the symbols of the pattern, and deducing the number of instances of each symbol in the pattern, as well as the proper order of the symbols within the pattern.

We create a DFA for each cyclic pattern (sub-channel) in a PLC-HMI channel. Then, for each PLC-HMI channel, a *Statechart* is built out of the multiple DFAs and it is used for the network intrusion enforcement phase.

A separate contribution is an extended and more granular representation of meta-data of the Siemens S7-0x72 SCADA protocol. Using knowledge from new versions of a Wireshark dissector [33] for the proprietary protocol we were able to refine the symbol construction by including more meta-data in each symbol. This resulted in four times more distinct symbols than in our previous analysis, allowing tighter representation of the normal SCADA traffic by our Statechart model.

We evaluated our solution on long traces from two production ICS: one using the Siemens S7-0x72 protocol and the other using Modbus. We also stress-tested our algorithms on a collection of synthetically-generated traces that simulated multiplexed ICS traces with aggressive levels of symbol uniqueness and time overlap.

The resulting Statecharts modeled the traces with an overall median false-alarm rate as low as 0.16% on the synthetic datasets (just above the optimal rate of 0.0042% achieved by the ideal Statechart), and zero false-alarms on production S7-0x72 traffic. Moreover, the spectral analysis Statecharts consistently outperformed the previous combinatorial Statecharts, exhibiting significantly lower false alarm rates and more compact model sizes. This validates the feasibility of the Statechart model of SCADA traffic which provides a solid basis for practical anomaly detection systems.

# 2. PRELIMINARIES

## 2.1 Adversary model

In this work we assume the existence of a semantic adversary who has unrestricted physical access to the SCADA network and thus has nearly complete control of the communications channel between the HMI and the PLCs. Our underlying threat model is based on the Dolev-Yao threat model [11]: The adversary may overhear and intercept all traffic regardless of its source and destination. The adversary can inject arbitrary packets with any source and destination addresses. Consequently, the adversary can also replay previously overheard messages. In particular the adversary can take over the HMI and issue control messages. The objective of the adversary is to manipulate the SCADA network to achieve an impact on the physical world.

Currently, most SCADA protocols do not include cryptographic algorithms such as ciphers and hash functions. Our adversary model assumes that if and when such security measures shall be deployed, their associated cryptographic keys will be known to (or can be broken by) the adversary. However, the adversary will be limited by the cryptographic methods employed by the communicating hosts. Hence, the adversary will not be able to subvert the cryptographic algorithms. We similarly require that in the presence of secure SCADA protocols, our NIDS will be configured with the necessary cryptographic keys so it would be able to decrypt the examined traffic.

We further assume that the adversary has in-depth knowl-

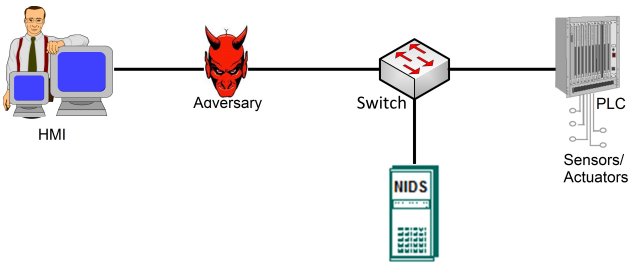


**Figure 1: Placing the Network Anomaly Detection System in a SCADA network**

edge of the architecture of the SCADA network and the various PLCs as well as sufficient knowledge of the physical process and the means to manipulate it via the SCADA system. Thus the adversary has the ability to fabricate messages that would result in real-world physical damage.

One example of a semantic adversary is described by Fovino et al. for a system with a pipe in which high pressure steam flows [14]. The pressure is regulated by two valves. An attacker capable of sending packets to the PLCs can force one valve to complete closure, and force the other to open. Each of these ICS commands is perfectly legal when considered individually, however when sent in an abnormal order they bring the system to a critical state. Another example [24] shows an attack scenario where a system-wide water hammer effect is caused. A fluid in motion is forced to stop or change direction suddenly, resulting in pressure surge or wave propagation in the pipe. The water hammer is caused simply by opening or closing major control valves too rapidly. This can result in a large number of simultaneous main breaks.

Fundamentally these attacks work by injecting messages into the communication stream, possibly legitimate messages, on an attacker-selected pattern and schedule. Hence a good anomaly detection system needs to model not only the messages in isolation but also their sequence and timing.

In our model the sensor would be located in the network segment where it can passively monitor traffic that was already modified by the adversary and just before the PLC as illustrated in Figure 1. The sensor is not located inline so it does not affect the normal network operation (e.g., port mirroring or a similar mechanism is used to instruct the switch to send copies of the network traffic to the anomaly detection system).

Note that our anomaly detection approach does not distinguish between malicious events and faulty events.

## 2.2 The GW model

The GW model [15] was originally developed and tested on Modbus traffic and later extended to other protocol suites such as various S7 flavors [18, 19]. Modbus is a simple request-response protocol widely used in SCADA networks. S7 is a family of proprietary protocols developed by Siemens, for the same purpose as Modbus, but with extended capabilities and more complex features. The Siemens PLCs are the predominant control devices in the control market.

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). In a Mealy DFA the output may depend on both the values of state and input variable. The GW' DFA symbols represent the

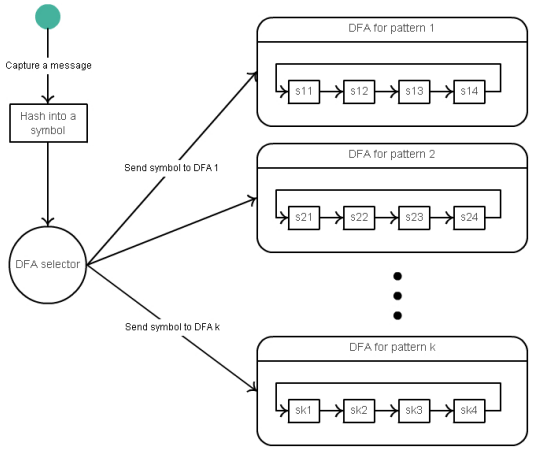


**Figure 2: A Statechart DFA model**

SCADA protocol messages' meta-data (commands and variable identifiers). The DFA represents the precise order of the symbols in the cyclic pattern. The GW model suggests a network anomaly detection system that comprises two stages: An unsupervised learning stage, and an enforcement stage. In the learning stage a fixed number of messages is captured, the pattern length is revealed, and the DFA is built for each HMI-PLC channel. The learning assumes that the sniffed traffic is benign. In the enforcement stage, traffic is monitored for each channel (according to its DFA), and proper events are triggered.

## 2.3 Modeling SCADA traffic with Statecharts

As hypothesized by [8], modern HMIs employ thread-based architecture, e.g., this is how the Afcon's Pulse HMI [1] is built. While each thread is responsible for certain tasks (e.g., controlling access to a range of registers on a PLC), the threads run concurrently with different scheduling frequencies, yet share the same network connections hence they produce multiplexed cyclic traffic patterns.

Attempting to model multiple cycles by a single DFA produces a very large, unwieldy model, as shown by [15]. Its normal pattern consists of many repetitions of the faster scan cycles followed by only few repetitions of the slow cycles. Such a pattern is also inaccurate since the slow cycles do not always interrupt the faster cycles at the same point, and while the slow patterns are active, symbols from all patterns are interleaved. Hence, in [19] we suggested a model that is based on a *Statechart* [17], a formalism that is more descriptive than a basic DFA.

The *Statechart* allows more accurate modeling of the traffic produced by modern HMIs (with the PLC's responses). The periodic traffic pattern driven by each thread in the HMI is modeled by its own DFA within the *Statechart* (see Figure 2). The *Statechart* also contains a DFA-selector to switch between DFAs. During the enforcement stage, each DFA in the *Statechart* maintains its own state, from which it transitions based on the observed symbols (messages). The DFA-selector's role is to send each of the input symbols to the appropriate DFA.

We identified two different scenarios that characterize the *Statechart*'s input-stream. In the simpler scenario each of the cyclic patterns of the input-stream consists of distinct symbols. In the more complex scenario there are symbol overlaps between different patterns within the input-stream.

| # | Len | Unq | Prd |
|---|---|---|---|
| **1** | 6 | 6 | 300 |
| | 4 | 4 | 950 |
| **2** | 6 | 6 | 300 |
| | 4 | 4 | 950 |
| **3** | 6 | 4 | 300 |
| | 4 | 1 | 400 |
| **4** | 6 | 4 | 300 |
| | 4 | 2 | 950 |
| **5** | 10 | 9 | 300 |
| | 4 | 2 | 600 |
| | 4 | 3 | 200 |
| **6** | 10 | 7 | 300 |
| | 10 | 7 | 950 |
| | 10 | 7 | 2000 |
| **7** | 10 | 8 | 300 |
| | 8 | 7 | 350 |
| | 10 | 9 | 400 |
| **8** | 10 | 8 | 300 |
| | 8 | 7 | 850 |
| | 10 | 9 | 1300 |
| **9** | 10 | 7 | 300 |
| | 8 | 4 | 350 |
| | 10 | 8 | 400 |
| **10** | 6 | 3 | 300 |
| | 4 | 2 | 350 |
| | 6 | 2 | 400 |
| **11** | 10 | 8 | 250 |
| | 4 | 2 | 650 |
| | 6 | 4 | 1100 |
| | 8 | 7 | 420 |
| **12** | 6 | 4 | 250 |
| | 4 | 4 | 350 |
| | 10 | 9 | 550 |
| | 8 | 7 | 420 |
| **13** | 10 | 9 | 300 |
| | 4 | 2 | 600 |
| | 4 | 2 | 200 |
| | 6 | 3 | 350 |

**Table 1: Overview of the sets of sequences used to generate the synthetic datasets. Each row shows the sequence length, the number of unique symbols in the sequence, and its period-time in msec units.**

In the learning-phase we need to construct a *Statechart* for each specific HMI-PLC channel, given a captured stream of symbols from the multiplexed channel.

## 2.4 Combinatorial Construction of the Statechart

In [20] we suggested a method to learn each of the multiplexed cyclic patterns even in cases when there is symbol overlap between different patterns. The learning algorithm starts with building a DTMC graph from the traffic stream. It then splits the symbol stream into sets, one per multiplexed cycle, based on symbol frequencies and node degrees in the DTMC graph. In the next step it creates a sub-graph for each cycle, and extracts Euler cycle alternatives for each sub-graph. The resulting patterns are then merged to create alternative *Statechart* DFAs of which the best is selected.

### 2.4.1 Testing the Construction with Real Data

The *Statechart* that was constructed as described above was tested using real SCADA traffic of the S7-0x72 protocol which has an asynchronous "subscribe-notify" mode that produces multiplexed cycles. For this protocol 64-bit hash symbols were calculated (using knowledge from the basic Wireshark dissector available at the time) out of 17-26 bytes taken from 11-17 meta-data fields of typical S7-0x72 packets. The S7-0x72 Statechart exhibited no overlapping symbols between DFAs and exhibited a false positive rate of 0.11%.

### 2.4.2 Stress Testing with Synthetic Data

In order to further test the model in more complicated scenarios, we stress-tested it with 13 different scenarios of SCADA traffic with various combinations of patterns, unique symbols per pattern, and frequencies (see Table 1 for details of the tested scenarios).

We implemented a multi-threaded generator, where each of the threads simulates an HMI thread transmitting a cyclic pattern of SCADA commands. Each simulated thread has a pattern $P$ of symbols, and a frequency $f$. Every $1/f$ msec the thread wakes up and emits the pattern $P$ as a burst, at a 1-msec-per-symbol rate, and returns to sleep. The thread's true timing has a jitter caused by the OS scheduling decisions. Further, when multiple threads are active concurrently then their emitted symbols are arbitrarily serialized.

The 13 generated scenarios vary the number of patterns, the number of unique symbols per pattern, and their frequency. The simpler scenarios (1-4) have 2 patterns each, while the most complex ones multiplex 4 patterns. Table 1 shows the parameters of the scenarios.

In [20] we showed that the combinatorial-construction statechart was successfully learned in 12 of the 13 scenarios. The average percentage of detected abnormals (false alarms) over the 12 scenarios was 4.3%.

Although the detected abnormal rate using the combinatorial construction is better than the 7% average rate of abnormal detection achieved by the Naive DFA over the same dataset, we see that the combinatorial method for Statechart construction leaves room for improvement especially when we compare this figure to the 0.94% rate that can be achieved by an Ideal *Statechart* on the synthetic datasets.

### 2.4.3 The need for better channel splitting

We observed that the major reason for the false alarms was the inaccuracy of channel splitting to sub-channel (pattern) components caused by the strict combinatorial requirements that characterize Euler cycles. Hence, in this paper, we suggest a totally different Statechart construction algorithm, based on spectral analysis. As we shell see, its performance is superior to that of the combinatorial approach, and close to that of the optimal Statechart.

## 2.5 Basics of Spectral Analysis

In the next sub-sections, we provide a brief introduction to the Fourier decomposition, which we use in the new learning method that we introduce in this paper.

### 2.5.1 The Fourier Transform

The normalized Discrete Fourier Transform (DFT) of a sequence $x(n), n = 0, 1...N-1$ is a vector of complex numbers $X(f)$:

$$X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}, \quad k = 0, 1...N-1 \quad (1)$$

where the subscript $k/N$ denotes the frequency that each coefficient captures. Since we are dealing with real signals, the Fourier coefficients are symmetric around the central one (the complex conjugate of their symmetric). The Fourier transform represents the original signal as a linear combination of the complex sinusoids $s_f(n) = \frac{e^{j2\pi fn/N}}{\sqrt{N}}$ . Therefore, the Fourier coefficients record the amplitude and phase of these sinusoids, after signal $x$ is projected on them.

### 2.5.2 Power Spectral Density Estimation using the Periodogram

In order to discover potential periodicities of a time-series, one needs to examine its power spectral density (PSD or power spectrum). The PSD indicates the signal power at each frequency in the spectrum. Since period is the inverse of

frequency, by identifying the frequencies that carry most of the energy, we can also discover the most dominant periods.

A well known estimator of the PSD is the periodogram. The periodogram P is a vector comprised of the squared magnitude of the Fourier coefficients. Consequently it can be computed using the DFT of a sequence (e.g., using the Fast Fourier Transform (FFT) for execution in $O(NlogN)$ time).

# 3. LEARNING CYCLIC PATTERNS VIA SPECTRAL ANALYSIS

In this section we describe our approach for learning the normal cyclic traffic of a multiplexed SCADA system and representing it as a Statechart.

Motivated by the work of Barbosa et al. [4], we chose to base our Statechart construction on spectral analysis. However, we take the idea much further: instead of looking only at the TCP packet sizes and arrival times, we apply deep packet inspection, and model the ICS protocol semantics in detail.

Broadly speaking, our approach works as follows. We start by treating the captured trace of SCADA symbols (messages) as a binary signal with two levels (0 and 1), where '1' at time $t$ indicates the presence of a message in the HMI-PLC channel at time $t$. Next we calculate the Fourier transform of the signal and its periodogram, thereby moving to the frequency domain. From the periodogram we identify the dominant periods of the signal: each dominant period will correspond to a cyclic SCADA symbol pattern (and ultimately, to a Statechart DFA). Returning from the frequency domain to the time domain of the trace, we then associate each symbol instance in the trace to one of the dominant periods. We treat the subsequence of symbol instances that is associated with a period as a SCADA sub–channel, and construct a GW–model DFA from it. The collection of DFAs then comprises the full Statechart.

## 3.1 The signal

The learning starts by collecting a pre-configured amount of $M$ sequential packets (the *learning window*) from each HMI-PLC channel, covering a span of $T$ seconds. We divide the collection period of each channel into time-segments of a chosen duration $\Delta t$ $\mu$sec. We define the Frequency of Artificial Sampling, $F_{AS} = 1/\Delta t$ and the number of samples in the learning window, to be $N = T/\Delta t$. Note that $\Delta t$ should be larger than the time it takes to transmit a message: when the communication is over 100 Mbps Ethernet, transmitting a 500 byte message takes $\approx 40\mu$sec, so setting $\Delta t \geq 1000\mu$sec is sufficient.

We analyze the communication activity of each HMI-PLC channel and store it as binary-valued signal $x[i]$: for every sampled interval $\Delta t_i$, we set $x[i] = 1$ if at least one packet belonging to the particular HMI-PLC channel occurs during this interval, and $x[i] = 0$ otherwise.

## 3.2 Periodicity learning

Based on Barbosa et al. [5] interviews with operators, we learn that SCADA applications do not normally perform polling faster than once per second. However in [15] the authors identified scan periods of 22–166msec. Hence we chose to set the value of $\Delta t$ to 1000$\mu$sec. By Nyquist's fundamental theorem the minimum period we are able to observe, 2msec, is more than sufficient to capture the periodicity of the SCADA traffic.

Once we have the signal $x$, we can carry out a Fourier analysis to obtain the signal's spectrum. To compute the FFT of $x$ efficiently, we adjust the length of $x$ to the maximum power of two value that is less than or equal to $N$. We then apply the Cooley-Tukey algorithm for FFT that is tailored to the factors of the sequence length to obtain the spectrum and periodogram of the signal.

Given the periodogram, we need to identify the dominant frequencies, and hence the dominant periods: naturally these would be the frequencies with largest powers (the ones that have the highest magnitude and correspond to the tallest peaks of the periodogram). However, there are three issues we need to address:

1. Harmonics: if we identify a dominant period with a frequency $f$ we typically also find periods with the frequencies $2 \cdot f, 3 \cdot f, ...$ as dominant. Therefore for every dominant frequency we need to eliminate all its integral multiples.
2. Spectral leakage (i.e., aliases of the original spectral component): the spectrum introduces "parasitic" frequencies, of low magnitude, that are not integer multiples of the DFT bin width, and are dispersed over the entire spectrum. For example spectral leakage might occur as a result of the signal sampling. Hence we need to eliminate the frequency components of low magnitude (the spectral leakage) by setting a threshold. A proper threshold should minimize the number of false alarms (i.e., non-dominant periods that are classified as dominant).
3. Adjacent frequencies: The sampled signal naturally has frequency jitter, caused by networking effects and finer grain thread scheduling in the HMI. Typically a dominant frequency in the spectrum is observed together with a batch of several adjacent frequencies, all of which pass the threshold. We need to select the peak frequency among each batch.

## 3.3 Detecting Dominant Periods

We use the periodogram to automate the extraction of dominant periods (peaks). Our automatic processing of a periodogram starts by selecting the values and locations of local maxima (peaks), and discarding the others. Then, we follow the approach of [32] to set the threshold. The authors assume a canonical model of a non-periodic time-series where a sequence of points are drawn independently and identically from a Gaussian distribution. The magnitudes of the coefficients of the DFT are distributed according to an exponential distribution where $\lambda$ is the inverse of the average power (denoted by $\mu$) of the peaks. Let the Threshold of Frequency Energy ($T_{FE}$) be a configurable value signifying the percentage of $\mu$, below which we omit periodogram frequencies. Let $X$ be the periodogram in the model of [32]. Then the cumulative distribution function is:

$$P(X \geq T_{FE}) = 1 - \int_0^{T_{FE}} \lambda e^{-\lambda X} \, dX \; = e^{-\lambda T_{FE}} \qquad (2)$$

Let $P' = 1 - P$. $P'$ represents the confidence probability that the returned periods will be significant. Under these conditions, [32] recommend to set $P$ to a certain low value and calculate the derived $T_{FE}$:

$$T_{FE} = -\mu \cdot ln(P) \qquad (3)$$

**Algorithm 3.1:** DETECTDOMPERIODS$(x, p, T_{FU})$

**comment:** $x$ - vector of binary-valued signal
$(1-p)$ - confidence prob. for dominant periods
$T_{FU}$ - configurable threshold of freq. unification

**main**
$n \leftarrow x.size$
$X \leftarrow FFT(x)$
$s \leftarrow 0$
$cnt \leftarrow 0$
$prev \leftarrow \|X[0]\|$
$crnt \leftarrow \|X[1]\|$
**for** $i \leftarrow 2$ **to** $x.size/2 - 1$
**do** { $next \leftarrow \|X[i]\|$
**if** $(prev \leq crnt)$ **and** $(crnt > next)$
**then** { $peaks[cnt] \leftarrow crnt$
$s \leftarrow (s + crnt)$
$cnt \leftarrow (cnt + 1)$
$prev \leftarrow crnt$
$crnt \leftarrow next$
$avgPeak \leftarrow (s/cnt)$
$T_{FE} \leftarrow (avgPeak * (-ln(p)))$
$j \leftarrow 0$
**for** $i \leftarrow 0$ **to** $peaks.size - 1$
**do** { **if** $(peaks[i] \geq T_{FE})$
**then** { $tPks[j] \leftarrow (n/i)$
$j \leftarrow (j + 1)$
**for** $i \leftarrow 0$ **to** $tPks.size - 1$
**do** { **for** $j \leftarrow 0$ **to** $i - 1$
**do** { $maxT \leftarrow MAX(tPks[i], tPks[j])$
$minT \leftarrow MIN(tPks[i], tPks[j])$
$k \leftarrow maxT/minT$
**if** $(|k - \lfloor k \rceil| < T_{FU})$
**then** $tPks.remove(minT)$
**return** $(tPks)$

Once we filter out frequencies below $T_{FE}$, we have a list of dominant period candidates. Next we want to keep only the first harmonics (the original periods), i.e., to filter out the higher harmonics (periods with frequencies that are positive integer multiple of the frequencies of the original periods).

For each pair of periods, with frequencies $f_1, f_2$ and $f_1 > f_2$, we discard $f_1$ if it is (or is almost) an integer multiple of $f_2$. For this purpose we define a Threshold of Frequency Unification ($T_{FU}$). Let $k = f_1/f_2$. Then if $|k - \lfloor k \rceil| < T_{FU}$ we discard the first period. We arbitrarily set the $T_{FU}$ value to 4%. Algorithm 3.1 shows pseudocode for the Dominant Period detection algorithm.

## 3.4 Finding the pattern symbols

Algorithm 3.1 returns the set of dominant periods - in units of time, for every HMI-PLC channel. Now we need to find the *symbols* of each of its cyclic patterns. For a given channel, let $D$ denote the set of dominant periods. We do the following (see Algorithm 3.2):

1. Find the set of distinct symbols $\sigma$, by examining the channel's corresponding (learning window) trace.
2. For each distinct symbol $s \in \sigma$ we:
   - Create a signal $y_s$ representing the occurrence of $s$: for every sampled interval $\Delta t_i$, we set $y_s[i] = 1$ if at least one instance of $s$ (belonging to the particular HMI-PLC channel) occurred during this interval, and $y_s[i] = 0$ otherwise.

**Algorithm 3.2:** FINDPTRNSYMLS$(D, st, T_{SM})$

**comment:** $D$ - frequencies of dominant periods
$st$ - captured symbols and their timestamps
$T_{SM}$ - configurable threshold of symbol multitude

**main**
$\sigma \leftarrow \{s | s \in st\}$
$sIdx \leftarrow 0$
**for each** $s \in \sigma$
**do** { **for** $i \leftarrow 0$ **to** $st.size - 1$
**do** { **if** $(st[i].contains(s)$
**then** $y_s[i] \leftarrow 1$
**else** $y_s[i] \leftarrow 0$
$fIdx \leftarrow 0$
**for each** $f_d \in D$
**do** { $Y_{d,s} \leftarrow DFT(f_d, y_s)$
$m[fIdx][sIdx] \leftarrow \|Y_{d,s}\|$
$fIdx \leftarrow (fIdx + 1)$
$sIdx \leftarrow (sIdx + 1)$
**for each** $m_d \in m$
**do** { $trshld \leftarrow (m_d.max * T_{SM})$
$sIdx \leftarrow 0$
$min \leftarrow \infty$
**for each** $sM \in m_d$
**do** { **if** $(sM > trshld)$
**then** { **if** $(sM < min)$
**then** $min \leftarrow sM$
**else** $m_d[sIdx] \leftarrow 0$
$sIdx \leftarrow (sIdx + 1)$
**for** $i \leftarrow 0$ **to** $m_d.size - 1$
**do** $m_d[i] \leftarrow \lfloor m_d[i]/min \rceil$
**return** $(m)$

**comment:** $m$ - $\forall(s, d) | s \in d, d \in D$ - stores:
$s$ magnitude; then reused to store $\#\ of\ s\ in\ d$

   - Compute the magnitude value $m_{d,s}$ of $s$ for each dominant period $d \in D$ of the HMI-PLC channel. For that, calculate the Fourier coefficient of the dominant period: $Y_{d,s} = DFT(f_d, y_s)$ for the frequency $f_d$ and the complex signal $y_s$ using equation (1). Then $m_{d,s} = \|Y_{d,s}\|$ where $\|.\|$ denotes the magnitude of a complex number.
3. For each dominant period $d$ we find the maximum magnitude of its symbols $MaxMg_d$. Let $T_{SM}$ be a configurable Threshold of Symbol Magnitude. We then filter out symbols with low magnitude: we assign to period $d$ all the symbols $s$ for which $m_{d,s} \geq MaxMg_d \cdot T_{SM}$. We arbitrarily set the $T_{SM}$ value to 5%. Importantly, a symbol $s$ may be assigned to several periods.
4. Note that symbol $s$ can appear multiple times in the period $d$. For each cyclic pattern $d$ we want to find the number of occurrences of every symbol $s$. For that, we first find the minimum magnitude of its symbols $MinMg_d$. We assume that the $MinMg_d$ corresponds to a symbol with a single occurrence in the pattern $d$. Then, the number of occurrences of any symbol $s \in d$ is given by: $\lfloor m_{d,s}/MinMg_d \rceil$.

Note that in Algorithm 3.1 we used the FFT algorithm with complexity $O(N \cdot logN)$, while in Algorithm 3.2, since we already know the frequency $f_d$, then for a cycle of $C$ symbols, we directly calculate the $C$ Fourier coefficients, giving a complexity of $O(C \cdot N)$, where $C$ is typically much smaller than $logN$.

### 3.5 Finding the symbol order within the pattern

Given a cycle $d$ with cycle time $CT_d$, algorithm 3.2 gives us the set $\sigma_d$ of symbols that belong to the cycle. In the simple case, in which the symbols of different cycles are distinct ($\sigma_d \cap \sigma_k = \emptyset, \forall k \neq d$), finding the order of symbols within the cycle is easy: filter the original trace to extract only the symbols in $\sigma_d$, and construct the cyclic pattern directly. This method works well even if a symbol $s \in \sigma_d$ appears multiple times in the pattern. However, in the complex case in which a symbol $s$ belongs to several cycles, we need to filter the trace more carefully.

At this stage we know the symbols (and their counts) per cyclic pattern of every HMI-PLC channel. Now we need to find the proper order of the symbols within each cyclic pattern. For each cyclic pattern we filter the packets (symbols) that belong to the particular cyclic pattern out of the collected packets. In case a symbol is shared between multiple patterns of the channel then for each instance of such symbol we need to determine to which particular pattern does the symbol instance belong.

Our decision on the mapping of symbol instances to patterns is according to a score we give to each of the alternative patterns to which a given symbol instance may belong to. The score for each symbol instance $S_i$ (where $i$ denotes the time slot number of that symbol at the symbol sequence) and pattern candidate with a cycle time $CT_d$ is calculated as follows:

For each symbol instance $s_i$ we define a sieve of all the potential points in time in the trace at which $s_i$ should appear if it belonged to cycle $d$. The sieve surrounds the potential points in time by a "slat" of $W$ timeslots to tolerate fluctuations. The sieve examines $C$ cycles back and $C$ cycles forward of the position of the symbol instance (at: $i \pm CT_d \cdot k \pm W$ where $1 \leqslant k \leqslant C$). In case the sequence ends before $(i + CT_d \cdot C)$ cycles or starts after $(i - CT_d \cdot C)$ cycles, we add corresponding cycles to the other side so that all together we examine $2 \cdot C$ cycles. The score is the fraction of the $2C$ "slats" in which $s_i$ appears. The pattern that gets the highest score is selected as the one to which the symbol instance belongs to.

Finally we split the collected symbols according to the different patterns each instance of symbol belongs to. We then learn the pattern of each of the resulting sequences using the GW learning model.

## 4. STATECHART ENFORCEMENT POLICIES

Before evaluating the performance of the spectral analysis Statechart construction, we return to the Statechart enforcement itself. As we described in Section 2.3 the *Statechart* includes a DFA selector whose role is to decide to which DFA to send each of the arrived input symbols. This decision is straightforward in the simple case when each of the cyclic patterns of the input-stream contains distinct symbols.

However in the more complex scenario some of the patterns overlap, and thus an arrived symbol may appear in multiple patterns of the same channel. In this case the DFA selector examines the current state of each of the alternative pattern-DFAs to select the best DFA the arrived symbol $s$ should be sent to. Note that as part of the learning, for each symbol in each DFA we record the average delay until the next symbol. Based on this information, by default the DFA selector assigns $s$ to the DFA whose predicted next-symbol-arrival time is closest to the current time. However there are cases where the DFA selector takes a different approach according to a pre-configured policy.

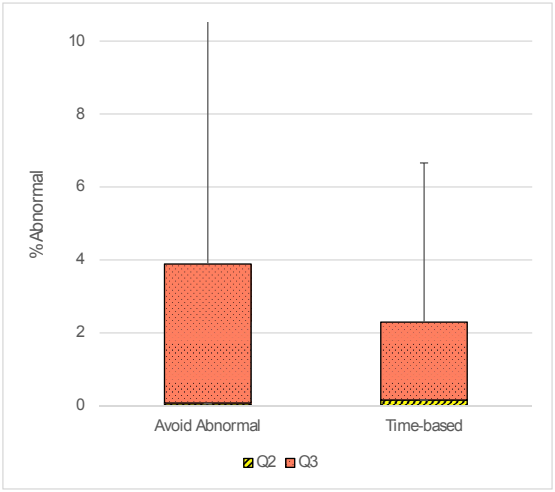


**Figure 3: Abnormal percentage for different enforcement policies (when the Spectral analysis model is used): diagonals represent the second quartile, the solid box represents the third quartile, and the "whisker" represents the maximum.**

Let $D_{crnt}$ denote the DFA that performed the last transition before the arrival of the current symbol $s$. There is a possibility that assigning the symbol $s$ to $D_{crnt}$ would lead to a *Normal* transition, while assigning $s$ to another candidate DFA would lead to a *Miss* or *Retransmit* transition. We examined 2 alternative policies for the best decision of the DFA selector in this case:

- *Avoid Abnormal*: Send $s$ to the current DFA in case it would transition, as a result, to a *Normal* state, while switching to any other DFA would lead to a *Miss* state or to a *Retransmit* state. This is the policy that was used in [19, 20]
- *Time-based*: Use only the time gap information to determine the DFA to which to assign the symbol.

We compared the performance of these 2 policies over the 13 generated scenarios of synthetic data (see Table 1) using the new Spectral analysis Statechart. The results are illustrated in Figure 3. The Figure shows that the median rate of detected abnormal symbols was very low regardless of the enforcement policy used (0.08% for the *Avoid Abnormal* and 0.16% for the *Time-based*).

However when we compare the third quartile rate and the maximum rate of detected abnormal symbols for the alternative policies, it is clear that the 2.29% and the 6.66% respectively achieved when the *Time-based* enforcement policy was applied is much better than the 3.89% and 23.61% that the *Avoid Abnormal* enforcement policy respectively yields.

Note that the synthetic traffic is all benign - so any alarm is a false alarm. Following this result we decided to use the *Time-base* policy for the reminder of our experiments.

## 5. EXPERIMENTS WITH SYNTHETIC DATA

In order to test our model in different scenarios, we reused the synthetic data from [19, 20] - recall Section 2.4.2.

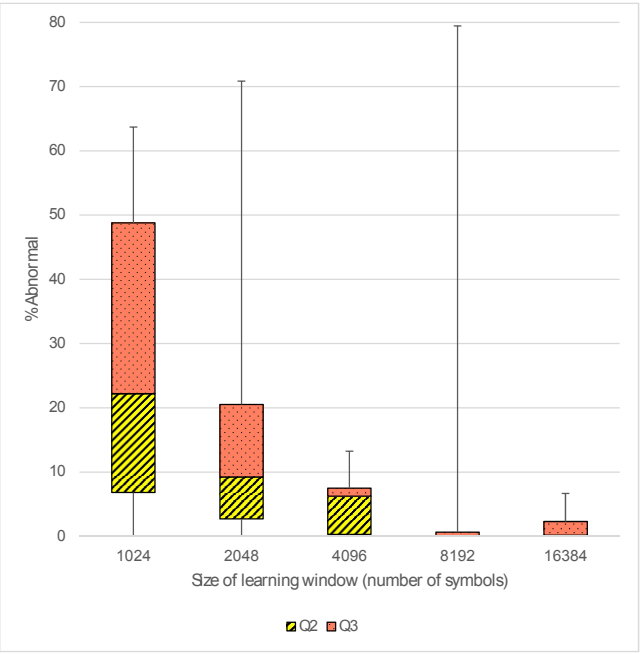


**Figure 4: Distribution of the abnormal rate for different learning window sizes.**

## 5.1 Calibrating the Learning window

We assumed that when we apply the Spectral analysis Statechart, the false alarm rate would decrease as we extend the size of the learning window. We verified this assumption using the 13 different synthetic scenarios. Figure 4 shows the results when we used window sizes of 1024, 2048, 4096, 8192, and 16384 symbols. The Figure clearly shows that the false alarm rate drops with larger learning window sizes. In the rest of the paper we used a window of N=16384 symbols.

## 5.2 Calibrating the Threshold

In order to set a proper $T_{FE}$ value, we tested several values for $P$ and calculated the derived $T_{FE}$ according to Equation 2. Figure 5 depicts the resulting abnormal rate as a function of the different probabilities of dominant periods. The best average abnormal rate is for confidence values $P' = 99.9975\%$. Note that setting the confidence too high is counter productive since correct periods are dropped. Thus setting the threshold $T_{FE}$ requires picking the largest possible confidence $P'$ (but no larger): Note the poor result when $P' = 99.999\%$.

## 5.3 Comparing Statechart Models

Once we calibrated the main parameters of the spectral analysis approach, we compared its performance to that of previous alternatives on the synthetic data. We compared to three possible alternatives:

1. A single naive DFA constructed using the method of [15].
2. A Statechart constructed using the combinatorial approach of [20] using a DTMC graph, from which Euler cycles are extracted.
3. The ideal Statechart: since the data is synthetic we know what the correct DFA is for each cyclic subchannel, hence we can construct the correct Statechart, as an unobtainable benchmark for comparison purposes.

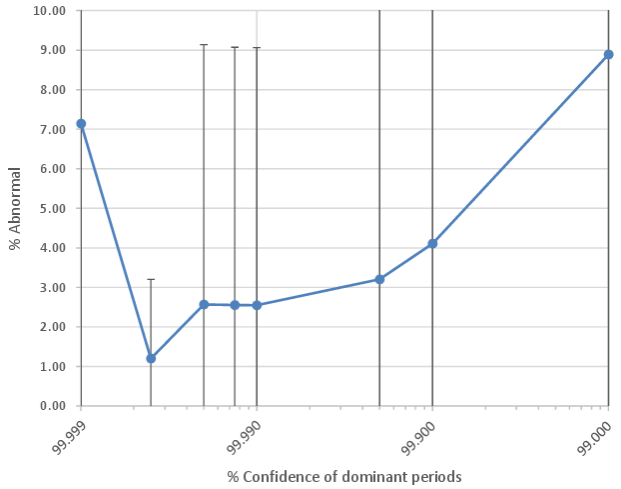


**Figure 5: Abnormals as a result of the used confidence of dominant periods. The points indicate the averages of % abnormals detected, and the bars are the standard deviations.**

| ID | Naive | Combinatorial | Spectral | Ideal |
|---|---|---|---|---|
| **1** | X | 10 | 10 | 10 |
| **2** | X | 10 | 10 | 10 |
| **3** | 36 | 8 | 10 | 10 |
| **4** | 222 | 94 | 10 | 10 |
| **5** | 22 | 15 | 18 | 18 |
| **6** | 130 | X | 212 | 30 |
| **7** | X | 28 | 28 | 28 |
| **8** | 86 | 28 | 28 | 28 |
| **9** | 46 | 29 | 108 | 28 |
| **10** | 74 | 16 | 16 | 16 |
| **11** | 74 | 37 | 78 | 36 |
| **12** | X | 28 | 28 | 28 |
| **13** | X | 27 | 24 | 24 |

**Table 2: Model sizes of the different scenarios of synthetic data for the alternative Statechart construction models. 'X' marks indicate an algorithm failure or a model with over 3000 symbols.**

Figure 6 shows the distributions of false alarm rates over the 13 synthetic scenarios, for the four models we compared. The Figure shows that the new spectral analysis approach is clearly superior to the previous Naive-DFA and DTMC-Euler constructions. With the Spectral-Statechart we observed a median false alarm rate of only 0.161% (in comparison to the 0.004% median false alarm rate of the ideal Statechart).

The Statechart model sizes are shown in Table 2. The table shows that the spectral analysis is also superior in term of model size: First, we can see that the naive method failed to create a reasonable-sized model in 5 of the 13 scenarios, and the combinatorial method failed to produce a Statechart in one scenario, whereas the spectral approach produced good statecharts everywhere. Further, we see that in 10 scenarios the spectral method produces an ideal-sized model. The largest gap versus the ideal is scenario 6, on which the combinatorial method failed. This scenario is especially difficult, with 3 cycles but only 70% symbol uniqueness (recall Table 1).

| Channel Id | A | B | C | D | E |
|---|---|---|---|---|---|
| Period time | 34 | 34 | 22 | 34 | 166 |
| Model size | 6 | 6 | 4 | 6 | 246 |

**Table 3: Model sizes of the Spectral-Statechart that were constructed for the different HMI-PLC channels of the Modbus traffic. The period times are given in $\mu$sec units.**

# 6. EXPERIMENTS WITH REAL TRAFIC

Due to the proprietary nature and potential sensitivity of SCADA operations, real SCADA network data is rarely released to researchers. An important aspect of this work is that we were able to examine and to analyze our new algorithms on traces we collected from production SCADA systems.

## 6.1 Modbus

Our first dataset is composed of traffic captured on a production Modbus network. The system monitors the electrical power utilization in the square kilometer campus of our university. The dataset includes 3.24 hours of Modbus traffic between the HMI and 5 PLCs (the HMI communicates separately with each of the PLCs). We denote the 5 HMI-PLC channels by A, B, C, D, and E. Each connection is maintained as a long-term TCP connection, which is immediately restarted on any disconnection.

We examined 9,398,302 Modbus packets of benign traffic. We believe that during this time the HMI exhibited a single-threaded execution per PLC. Hence the resulting Statecharts should all ideally consist of a single DFA.

Applying the naive GW model on this traffic resulted in detection of 9,327,465 (99.25%) "Normal" packets. In 4 of the channels (A-D) we observed very low rate of abnormals (0.0018%-0.0025%). Channel E exhibits 5.6% abnormals: upon inspection we discovered that this channel is indeed noisy, with many Modbus errors causing the PLC to send response packets with exception codes.

We then applied the spectral-based Statechart model. We set the confidence of the dominant period to 99.92%. The results were exactly the same as those achieved by the Naive model. Figure 7 shows the abnormal rate for the five channels. The model sizes for these channels (shown in Table 3) were also the same as the sizes constructed by the naive GW model.

We conclude that the spectral analysis performs extremely well on non-multiplexed single threaded Modbus traffic, despite the fact that it does not assume only a single-cycle exists.

## 6.2 S7-0x72

Our second dataset is composed of traffic captured on a production S7 network running the Siemens S7-0x72 protocol from a control network of a solar power plant. In this trace we observed a single channel between the HMI and a Siemens S7-1500 PLC. We observed both the request-response and the unique subscribe/notification communication patterns, hence the traffic is multiplexed. We used this dataset in [20].

### 6.2.1 *The need for meta-data of higher granularity*

Recall that modeling SCADA traffic for anomaly detection requires a representation of the traffic packets by symbols. Recently, based on an improved Wireshark dissector of the S7-0x72 protocol [33], we attained a better understanding of the protocol payloads and managed to classify additional fields of typical S7-0x72 packets as meta-data fields. These additional fields provide higher granularity of the S7-0x72 meta-data and thus a better symbol representation for the anomaly detection than the representation used in [20].

Separately, we noticed that S7-0x72 response packets include minimal meta data. The meaningful meta-data (variable IDs and types) appears only in the "request" message, and the response consists mostly of the returned data values, which are excluded from the hash function we use to create the symbol. Consequently, if we compute the response messages' symbols naively, then symbols associated with different response messages will be the same. Therefore in this paper we chose to combine the request message's meta-data into the corresponding response's symbol – by treating the request message's hashed symbol as a "field" in the response message, thereby hashing it into the response's symbol. We

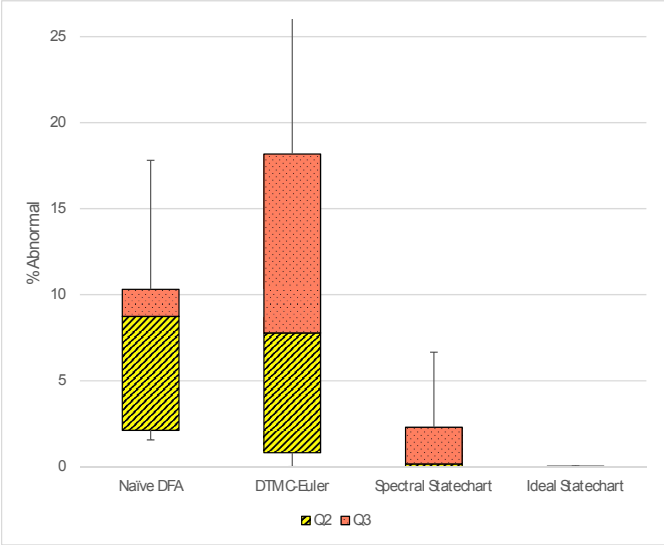


**Figure 6: Abnormal rate of different Statechart construction models for the synthetic data**

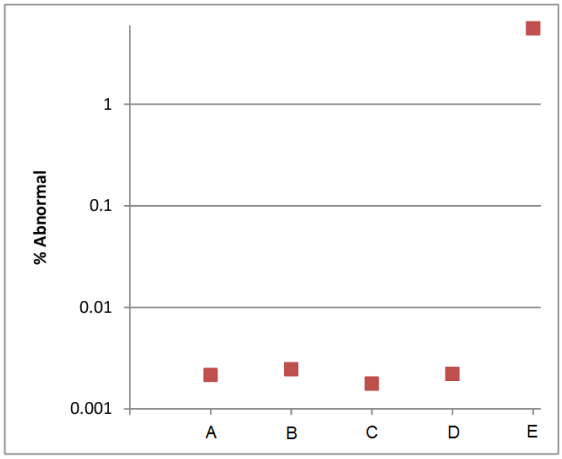


**Figure 7: Abnormal percentage for different Modbus channels (when the Spectral analysis model is used)**

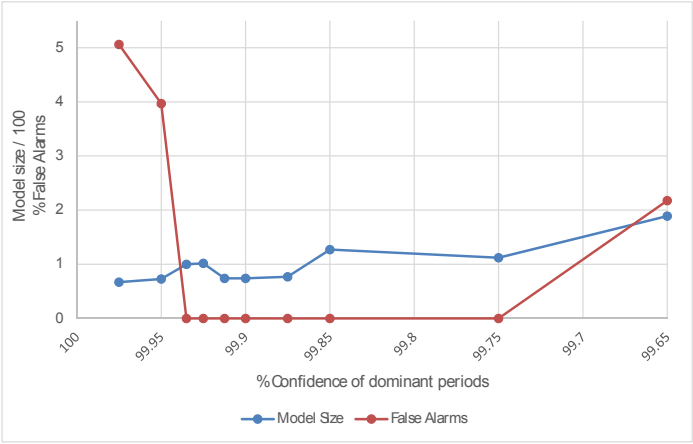


**Figure 8: Abnormals and model sizes as a result of the used confidence of dominant periods (after applying the Spectral-based Statechart on S7-0x72 traffic)**

are able to do this since the transaction Id field that appears in each request or response packet in the S7-0x72 protocol payload has the same value in the request packet and its associated response packet.

In this work, we calculated the 64-bit hash symbols out of 13-71 bytes (on average ≈24 Bytes) taken from 16-51 meta-data fields (on average ≈22 meta-data fields) of typical S7-0x72 packets. The meta-data we extracted for the learning and the enforcement of the model represents ≈36% of the fields of a typical S7-0x72 packet.

As a result of these improvements in the symbol calculation, the S7-0x72 dataset exhibits 13 distinct symbols (5 pairs of request-response messages and 3 singles of notification messages type). In comparison, in [20] we used a naive symbol calculation on the same dataset, and only observed 3 distinct symbols (1 pair of request-response and 1 notification). Hence, the new symbols allow a much more accurate and granular model of the traffic.

Figure 8 depicts both the false-alarm rate and the sizes of the learned model as a function of the probability of dominant periods. It shows that by configuring the $T_{FE}$ to a value corresponding to any probability between $5 \cdot 10^{-4} - 25 \cdot 10^{-4} (P' \in [99.75\% - 99.95\%])$ the enforcement of the spectral-*Statechart* model does not produce *any* false alarm on the S7-0x72 traffic.

# 7. CONCLUSIONS AND FUTURE WORK

In this paper we developed and applied a new Statechart construction method that is based on spectral analysis. Our Statechart DFA model is designed specifically for anomaly detection in ICS networks. The model includes a methodology for unsupervised learning of individual patterns of ICS traffic that is sometimes multiplexed, due to multi-threaded scheduling.

We suggest and evaluate a new approach for splitting of multiplexed input stream of a PLC-HMI channel into multiple sub-channels. Our algorithm then associates a set of symbols with each sub-channel, identifies the order of the symbols within each sub-channel, and creates the cyclic DFAs and the Statechart.

Our channel splitting algorithm performs very well even in the challenging cases, when we don't know in advance how many sub-channels exist, when there is symbol repetition within the same pattern, and when the sub-channels potentially have overlapping symbols.

We evaluated our solution on long traces from two production ICS: one using the Siemens S7-0x72 protocol and the other using Modbus. We also stress-tested our algorithms on a collection of synthetically-generated traces that simulated multiplexed ICS traces with varying levels of symbol uniqueness and time overlap. The resulting Statecharts modeled the traces with an overall median false-alarm rate as low as 0.16% on the synthetic datasets, and zero false-alarms on production S7-0x72 traffic. Moreover, the spectral analysis Statecharts consistently outperformed the previous combinatorial Statecharts, exhibiting significantly lower false alarm rates and more compact model sizes.

The Statechart DFA model has three promising characteristics. First, it exhibits very low false positive rates despite its high sensitivity. Second, it is extremely efficient: it has a compact representation, it keeps minimal state during the enforcement phase, and can easily work at line-speed for real-time anomaly detection. Third, its inherent modular architecture makes it scalable for protecting highly multiplexed ICS streams. The methodology for the automatic construction of the model, the characteristic of the model, and the validation of the model by our experiments suggest that this model can be very useful for anomaly detection in ICS networks.

We still need to test the algorithms' performance on additional real traces - and also test the Statechart's ability to detect true attacks.

# 8. ACKNOWLEDGMENTS

This research was supported in part by a grant from the Interdisciplinary Cyber Research Center at TAU.

# 9. REFERENCES

[1] Afcon Technologies. Pulse HMI software, 2015. [Online; accessed 24-Nov-2015].

[2] C. Alcaraz, L. Cazorla, and G. Fernandez. Context-awareness using anomaly-based detectors for smart grid domains. In *9th International Conference on Risks and Security of Internet and Systems*, volume 8924, pages 17–34, Trento, 04/2015 2015. Springer International Publishing, Springer International Publishing.

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

[4] R. Barbosa, R. Sadre, and A. Pras. A first look into SCADA network traffic. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 518–521, April 2012.

[5] R. Barbosa, R. Sadre, and A. Pras. Towards periodicity based anomaly detection in scada networks. In *17th IEEE Emerging Technologies Factory Automation (ETFA)*, pages 1–4, Sept 2012.

[6] L. Briesemeister, S. Cheung, U. Lindqvist, and A. Valdes. Detection, correlation, and visualization of attacks against critical infrastructure systems. In *8th*

*International Conference on Privacy Security and Trust (PST)*, pages 17–19, 2010.

[7] E. J. Byres, M. Franz, and D. Miller. The use of attack trees in assessing vulnerabilities in SCADA systems. In *Proceedings of the International Infrastructure Survivability Workshop*, 2004.

[8] M. Caselli, E. Zambon, and F. Kargl. Sequence-aware intrusion detection in industrial control systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 13–24, New York, NY, USA, 2015.

[9] C.-M. Chen, H.-W. Hsiao, P.-Y. Yang, and Y.-H. Ou. Defending malicious attacks in cyber physical systems. In *IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA), 2013*, pages 13–18, Aug 2013.

[10] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes. Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA Security Scientific Symposium*, pages 127–134, 2007.

[11] D. Dolev and A. C. Yao. On the security of public key protocols. Technical report, Stanford, CA, USA, 1981.

[12] N. Erez and A. Wool. Control variable classification, modeling and anomaly detection in modbus/tcp scada systems. *International Journal of Critical Infrastructure Protection*, 10(C):59–70, Sept. 2015.

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

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

[15] N. Goldenberg and A. Wool. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection*, 6(2):63–75, 2013.

[16] D. Hadziosmanovic, D. Bolzoni, P. H. Hartel, and S. Etalle. MELISSA: Towards automated detection of undesirable user actions in critical infrastructures. In *Proceedings of the European Conference on Computer Network Defense, EC2ND 2011, Gothenburg, Sweden*, pages 41–48, USA, September 2011. IEEE Computer Society.

[17] D. Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8(3):231–274, June 1987.

[18] A. Kleinmann and A. Wool. Accurate modeling of the siemens S7 SCADA protocol for intrusion detection and digital forensic. *JDFSL*, 9(2):37–50, 2014.

[19] A. Kleinmann and A. Wool. A statechart-based anomaly detection model for multi-threaded SCADA systems. In *10th Critical Information Infrastructures Security (CRITIS)*, pages 132–144, Berlin, Germany, Oct. 2015. Springer.

[20] A. Kleinmann and A. Wool. Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems. *arXiv preprint cs.CR/1607.07489*, 2016.

[21] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth. Internet-facing PLCs-a new back orifice. In *Blackhat USA 2015, Las Vegas, USA*, 2015.

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

[23] T. D. Maiziere. Die lage der it-sicherheit in deutschland 2014. Technical report, Bundesamt fur Sicherheit in der Informationstechnik, 2014.

[24] R. T. Marsh. Critical foundations: Protecting america's infrastructures - the report of the president's commission on critical infrastructure protection. Technical report, October 1997.

[25] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *Network, IEEE*, 8(3):26–41, 1994.

[26] P. A. Porras and P. G. Neumann. EMERALD: event monitoring enabling responses to anomalous live disturbances. In *1997 National Information Systems Security Conference*, Oct. 1997.

[27] M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX Conference on System Administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association.

[28] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *IEEE Security and Privacy (SP)*, pages 305–316, May 2010.

[29] R. Spenneberg, M. Brüggemann, and H. Schwartke. PLC-blaster: A worm living solely in the PLC. In *Black Hat Asia, Marina Bay Sands, Singapore*, 2016.

[30] K. A. Stouffer, J. A. Falco, and K. A. Scarfone. Guide to industrial control systems (ICS) security. Technical Report 800-82, National Institute of Standards and Technology (NIST), Gaithersburg, MD, May 2013.

[31] A. Valdes and S. Cheung. Communication pattern anomaly detection in process control systems. In *IEEE Conference on Technologies for Homeland Security (HST)*, pages 22–29. IEEE, 2009.

[32] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 131–142. ACM, 2004.

[33] T. Wiens. S7comm wireshark dissector plugin, January 2014. Available at: http://sourceforge.net/projects/s7commwireshark.

[34] D. Yang, A. Usynin, and J. Hines. Anomaly-based intrusion detection for SCADA systems. In *5th Intl. Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies*, pages 12–16, 2006.

[35] N. Ye, Y. Zhang, and C. Borror. Robustness of the markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, 53(1):116–123, 2004.