Monitoring behavior of process variable and anomaly detection in its time series data from modbus packet flow

Mustafa Amir Faisal

1 Introduction

Industrial Control systems' (ICSs') security is now a reality specially after Stuxnet [5] attack, where the speed of centrifuges of Iranian uranium enrichment plant is manipulated to operate outside of their operational range. This results a hardware damage. ICS, an instance of generic Cyber-Physical System (CPS) architecture, is organized in a layered hierarchy [9] where the highest layers focus with enterprise network and the lowest level focus on the direct interaction with the physical system. In lowest layer, actuators (pumps, valves, etc.) and sensors (thermometers, tachometers, etc.) communicate with a controller—PLC (Programmable Logic Control), a Remote Terminal Unit (RTU), or Intelligent Electronic Device (IED) via some time-critical communication protocols like Modbus, DNP3, Profinet, Ethernet/IP, MMS, etc. All of these protocols are used for various purposes in ICS communication networks.

In this project, we focus on the monitoring network traffic of PLCs, used Modbus, with other devices. This is done by extracting the update of process variables (represented by registers of PLC). PLCs act as the interface between a plant's operators and field devices (like valves, etc.). Thus, any state changes, including malicious command, have effect on the system. Thus, PLCs are ideal points in the ICS network to monitor their traffic to detect malicious activities. Modbus is a de facto communication protocol among PLCs, and various devices. It is a simple, robust, and royalty free ICS protocol. In Modbus, every PLC defines a memory map representing an internal table of process variables (a few thousands normally).

We use time series data generated from the update of process variable in a PLC. This project report is organized as follows: in section 2, we provide the concrete problem definition of the problem, we like to focus in this project. Section 3 includes theoretical background of our approach and evaluation metric for it. Experimental setup which consists of data set description, preprocessing of it, and experimental details is presented in section 4. In section 5, we will provide the results of our experiments. A precise discussion will be presented in section 6. 7 provides a precise related work of this project. The extension or future work will be provided in section 8. Finally, we conclude this report with conclusion in section 9.

2 Problem Definition

For monitoring the process variables of PLC, we consider a logical anomaly detection architecture. In this architecture, we consider y, u, r, v, and z as the communication channel between sensor and controller, controller and actuator, detection system and reconfiguration (response while attack is detected), between actuator and physical process (Plant), and between physical process and sensor respectively. Figure 1 shows this logical architecture. By providing the suffix of k, we mean particular control signal in the specific channel.

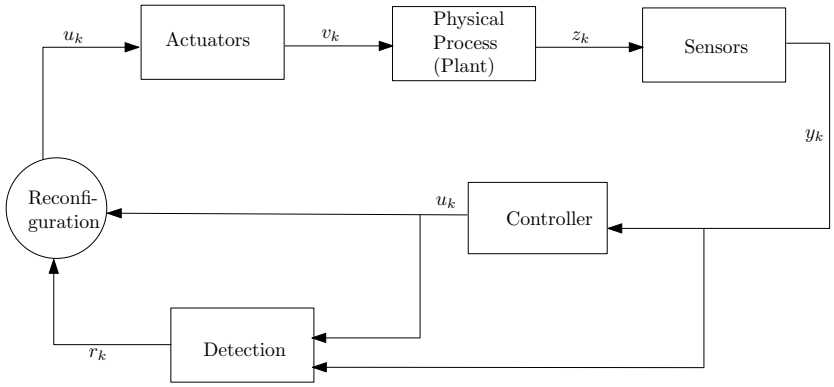


Figure 1: Logical Anomaly Detection Architecture. Under normal operating conditions, the actuation on the plant corresponds to the intended action by the controller: $v_k = u_k$, the variables of observation are the same as the sensor measurements reported back to the controller: $y_k = z_k$, and the controller generates a control action based on the correct control logic $\mathcal{K}()$ given y_k, i.e., $u_k = \mathcal{K}(y_k)$.

In this project, we are concern about the attack on channel y. See Figure 2 which provides the pictorial version of this attack. In this attack, the adversary deceive the controller about the real state of the plant. In our case, the process variable in PLC will be update by the manipulated or invalid sensor measurement.

To tackle this attack on manipulated sensor measurement, we will provide some schemes which can be implement in detection block in Figure 1 or 2. However, in our case, we do not know the input command u_k from controller. So, we have to use only output signal y_k from sensor.

Our aim in this project is to select appropriate model for describing the time series generated in a register (store process variable) in PLC. And evaluate two bad data injection detection methods using this model. Single and correlated signals are be combined with these detection techniques (we will discuss them in details later).

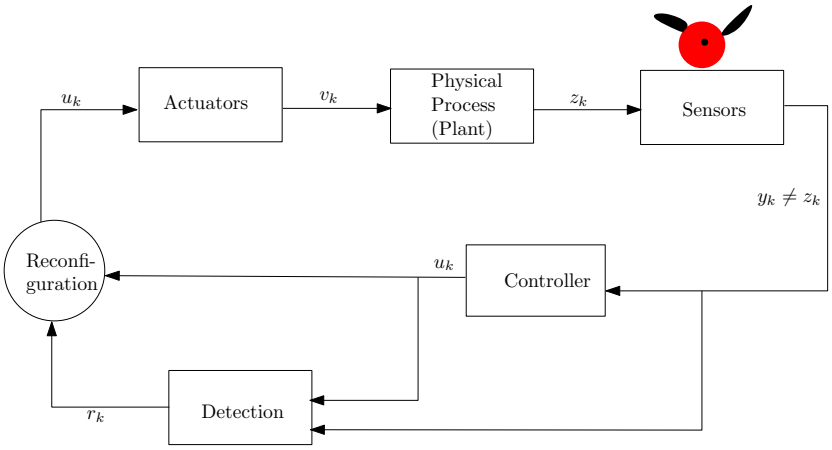


Figure 2: When one or more sensor signals are compromised (e.g. the sensor itself is compromised or the controller receives and accepts a sensor measurement from an untrusted entity) the sensor measurement used as an input to the control algorithm will be different from the real state of the measured variable $y_k \neq z_k$.

3 Methodology

3.1 Background

The detection block can be composed of various algorithms like 'residual generation' and the 'bad data detector'. This 'residual generation' algorithm requires a mathematical model of the system for prediction which normally achieves via 'system identification'.

In a straightforward way to detect anomaly is to compare the process behavior with a process model (predictor) describing the normal behavior. The difference of signals between the process and the model are called residuals. Thus the expected measurement, $\hat{y}_k$ generated by the model is compared with the actual sensor measurement, y_k and their difference is called the residual, r_k.

Now, the model used by the 'residual generation' algorithm can be developed by input/output (in our case output only) observation and this techniques is named as **system identification** [1].

System identification uses machine learning/statistical methods to build mathematical models of dynamical systems from observed input-output data or output-only data. In practice, manipulated input changes are made at discrete time intervals and measured output are available at discrete sample times. However, physical inputs and outputs are continuous; only the changes and measurements occur at discrete times. This leads us to focus on discrete autoregressive models which are commonly implemented in real application. As we have only output data (y_k), regression models like AR, ARMA, or ARIMA are a popular way to learn the correlation between observations. These model are non-seasonal model. We can use the seasonal versions of these models. A brief description of these models will be provided here.

AR (Autoregressive) model:

AR model provide a linear association between lagged (previous) observations. This guides to predict current observations from past observations.

Let y_t is present observation and $y_{t-1}, y_{t-2}, ..., y_{t-p}$ are past p number observations. AR provides a functional representation which predicts current observation, y_t from previous p, order or degree of AR, observations. The model is denoted as $AR(p)$

$$y_t = c + \sum_{i=t-i}^{p} \chi_i y_i + \epsilon_t \tag{1}$$

where c is constant and χ_i is co-efficient corresponds with each y_i observation. ϵ_t is white noise assumed uncorrelated innovation process with mean zero.

ARMA (Autoregressive and moving average) model:

Lets first define Moving Average (MA). If $\epsilon_{t-1}, \epsilon_{t-2}, ..., \epsilon_{t-q}$ are last q innovations (the difference between the observed value y_t and optimal for-casted value based on information prior to time t), ϵ_t is white noise like AR, and mean of time series is μ, then $MA(q)$ can be defined as follows:

$$y_t = \mu + \sum_{i=t-i}^{q} \theta_i \epsilon_i + \epsilon_t \tag{2}$$

Now, ARMA model is a combination of AR and MA, normally good when time series require higher order or degree. And can be defined, $ARMA(p, q)$ as follows:

$$y_t = c + \sum_{i=t-i}^{p} \chi_i y_i + \sum_{i=t-i}^{q} \theta_i \epsilon_i + \epsilon_t \tag{3}$$

ARIMA (Autoregressive integrated moving average) model:

ARIMA model fits for the time series which show non-stationary, that means, the mean, variance etc. of time series change over time. Non-stationary part can be changed to stationary by taking d differences, difference-stationary or unit root processes. $ARIMA(p, d, q)$ can be defined as follows:

$$\Delta^d y_t = c + \sum_{i=t-i}^{p} \Phi_i \Delta^d y_i + \epsilon_t + \sum_{i=t-i}^{q} \theta_i \epsilon_i \tag{4}$$

where $\Delta^d y_t$ denotes $d-$differenced time series.

Seasonality issue:

So far we discuss about non-seasonal models. However, there can be happened where patterns repeat successive time in time series. And this is known as seasonality in time series data. Multiplicative ARIMA model focuses seasonality in the data.

Now, if lag operator, L in polynomial notation, $L^i y_t = y_{t-i}$, then a time series with periodicity s has multiplicative ARIMA model, $ARIMA(p, d, q) \times (p_s, d_s, q_s)$ is give as:

$$\varphi(L)\Phi(L)(1-L)^d(1-L^s)^{d_s}y_t = c + \theta(L)\Theta(L)\epsilon_t \tag{5}$$

where $\varphi(L)$ is stable AR(q) model, defined as $\varphi(L) = 1 - \varphi_1 L - \cdots \varphi_p L^p$. $\Phi(L)$ can be defines as with p_s which is the seasonal part for AR. Similarly, with degree q, MA operator polynomial is $\theta(L) = 1 + \theta_1 L + \cdots + \theta_q L^q$ and its seasonal part with degree, q_s is $\Theta(L)$. Finally, $(1-L)^d$ and $(1-L)^{d_s}$ are non-stationary in observations for integrated parts.

Model and its Order or Degree Selection:

Now, all of these models may not fit for our data. We use Box-Jenkins methodology[2] for selecting the proper model for our data and the required degree.

At this point, we are interested about the bad data detection using this selected model with some statistical test. In this case, statistical test is an algorithm, $\mathcal{D}()$ take residuals, r_k as inputs and generate alarm in case of inconsistency. The decision can be done for each r_k by generating an "alert" on a process variable at any time during testing a batch of data while an observation deviates from the prediction. This leads to select a threshold τ.

If $|r_k| \leq \tau$, the no alert is generated. And if

$$|r_k| > \tau \tag{6}$$

an alert will be generated. A problem with this method, making a decision for each r_k, is the decision maker is memoryless, i.e., each time it makes a decision, it forgets the past history of potentially malicious activities. A possible solution can be taking the sum or average and check it against the threshold value. However, it can delay decision making and an optimized way needs to find out.

Change detection techniques like CUmulative SUM (CUSUM) can be used as an alternative. CUSUM assumes we have a probability model for our observations r_k. However, it in not practical and so, we focus on the non-parametric CUSUM (CUSUM without probability likelihood models). This approach basically a sum of the residuals. In this case the CUSUM statistic is defined recursively as $S_0 = 0$ and $S_{k+1} = (S_k + |r_k| - \delta)^+$ where $(x)^+$ represents $max(0, x)$ and δ is selected so that the expected value of $|r_k| - \delta < 0$ under hypothesis that δ prevents S_k from increasing consistently under normal operation. An alert is generated whenever the statistic is greater than a previously defined threshold

$$S_k > \tau \tag{7}$$

and as soon as an alarm is generated, the test is restarted with $S_{k+1} = 0$.

3.2 Evaluation Metric

In stead of using traditional measures like trade-off between true positive rate and false positive rate, we use a metric of the security of a detection algorithm: the maximum amount that an attack can deviate a target signal without being detected given a fixed period of time. The reason using this trade-off for the algorithm, it seems artificial as testing the true positive rate, an attack need to be generated like random attack [8, 3]. Thus, this approach is not adaptive and will not evade the detection algorithm. Thus, this leads many proposals have

lost interest from using true positive rate to evaluate their detection algorithms [4, 6].

We use the ideas from change detection theory where the evaluation of a statistical test is not done in terms of the probability of false alarm but on the average waiting time between false alarms, T_{FA}. More precisely, the expected time between false alarms $\mathbb{E}[T_{FA}]$.

3.3 Threat Model

We consider a greedy attacker that at each time step will try to maximize the effect it observes at the tnext time step. Let $f(\hat{y}_{k+1}, y^s_{k+1})$ be the predicted impact of the attack. The objective of the attacker is to fulfill the following optimization problem:

$$\max f$$
$$s.t. |r_k| \leq \tau$$

Our assumption is that the attacker has control over the sensor signal $y^s_k = y^a_k$ where y^a_k is the manipulated sensor signal lunched by the attacker.

Let t be the time at which attack is introduced. Thus, the objective of the attacker is to find an optimal y^{a*}_{t+1} that maximizes the future impact $f(\hat{y}_{t+1}, y^s_{t+1})$ at each $t_{next} > t$. In this way, the optimization problem reduced to for memoryless case:

$$y^{a*}_{t+1} = \max\{y^a_{t+1} : |y^a_{t+1} - \hat{y}_{t+1}| \leq \tau\}$$

Certainly, the solution to this problem corresponds to the case when $|y^a_{t+1} - \hat{y}^s_{t+1}|$. Then

$$y^{a*}_{t+1} = \hat{y}_{t+1} \pm \tau \tag{8}$$

Now in case of CUSUM,

$$y^{a*}_{t+1} = \max\{y^a_{t+1} : S_{t+1} \leq \tau\}$$

As $S_{k+1} = (S_k + r_k - \delta)$, we assume the optimal attack will be when $S_k = \tau$ that gives us:

$$y^{a*}_{t+1} = \hat{y}_{t+1} \pm (\tau + \delta - S_k) \tag{9}$$

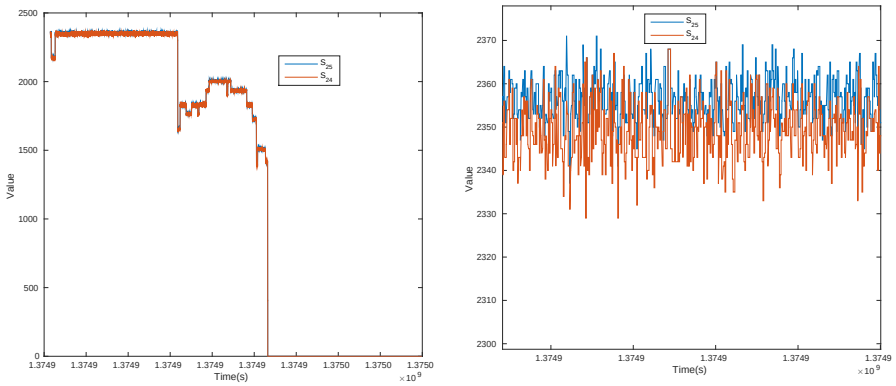
Noticeable, using the CUSUM, the effect of the attack is independent on τ as for a $t_{next} > t, S_k = \tau$ and $y^{a*}_{t+1} = \hat{y}_{t+1} \pm \delta$. And if $\delta < \tau$, the attacks are less effective than with the memoryless detection approach.

We will evaluate these two approaches with single and two most co-related signals.

4 Experimental Setup

4.1 Data Set

Data set is collected from real data captured from a water utility company for 24 hours which generate 166 pcap files (200 GB). All the pacap files are



(a) Correlated signals S_{25} and S_{24}

(b) Segment of correlated signals S_{25} and S_{24}

Figure 3: Highest correlated signals with $corr(S_i, S_j) > .9999$.

replayed with capture time and network analysis framework **Bro** is used to track the memory map of holding (read/write) registers and log the changes. From analysis with **Bro**, we found that i) 95% of transmission are Modbus packets and the rest 5% are from other non-ICS protocols like SNMP, SMB, DHCP, LDAP, NTP; ii) the trace captured 108 Modbus devices, of which one acts as central master, one as proxy to communicate with external network, and 106 are slave PLCs; iii) among the commands (master to slave PLCs) 74% are Read/Write Multiple Registers ($0x17$) commands, 20% are Read Coils (0x01) commands, and 6% are Read Discrete Inputs ($0x02$) commands; and iv) 78% of PLCs consist of 200 to 600 registers, 15% of PLCs consist of 600 to 1000, and 7% with more than 1000. **Pandas**, a Python Data Analysis Library is used to parse the log generated by Bro for extracting time series data for registers of each PLC. Each time series corresponds to a signal (s_k) in our experiments.

Now applying the characterization approach mentioned in [6], we classified the signals as 91.5% are constant, 5.3% are discrete, and 3.2% are continuous. In our case, we are interested only continuous data. As single PLC's time series data is important, we select a PLC (with IP address 172.16.11.229) randomly. And it has 135 registers which deal with continuous signals. From these 135 signals, we selected two most correlated ones (discussed in detail later).

4.2 Tools

AR and its corresponding variants have various implementations like System Identification and Econometrics toolbox in MathLAB, *stats* and *forecast* packages in R, etc. In this project we use System Identification toolbox in MatLAB for its simplicity.

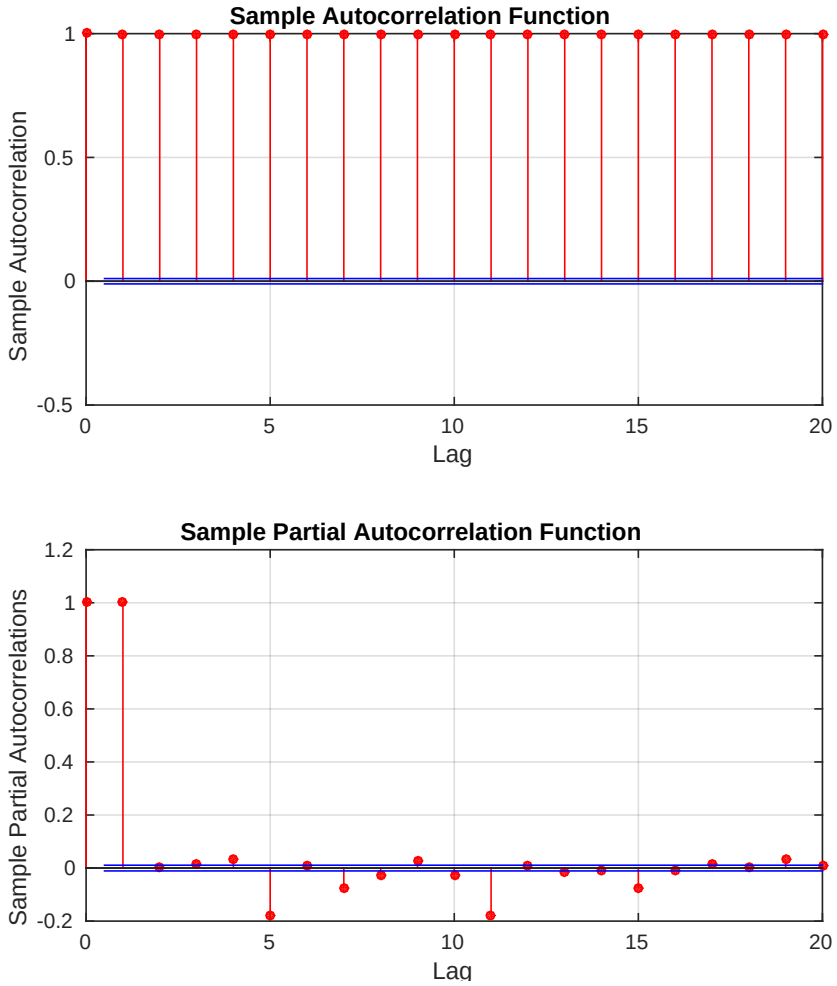


Figure 4: ACF and PACF for signal S_{25}.

4.3 Experiments

4.3.1 Signals Selection

As we have mentioned earlier, in our selected PLC (with IP address 172.16.11.229) has 135 continuous signals we selected most correlated signals.

We correlated signals by computing the correlation coefficients from an input matrix S whose rows are observations and whose columns correspond to different signals $s_1, s_2, \cdots, s_N$. The correlation coefficient is a normalized variant of the mathematical covariance function. For s_i and s_j, the correlation is

$$corr(s_i, s_j) = \frac{cov(s_i, s_j)}{\sqrt{cov(s_i, s_i)cov(s_j, s_j)}}$$

where $cov(S_i, S_j)$ denotes the covariance between S_i and S_j and correlation ranges between $-1 \leq corr(S_i, S_j) \leq 1$. We then calculate the *p-value* of the

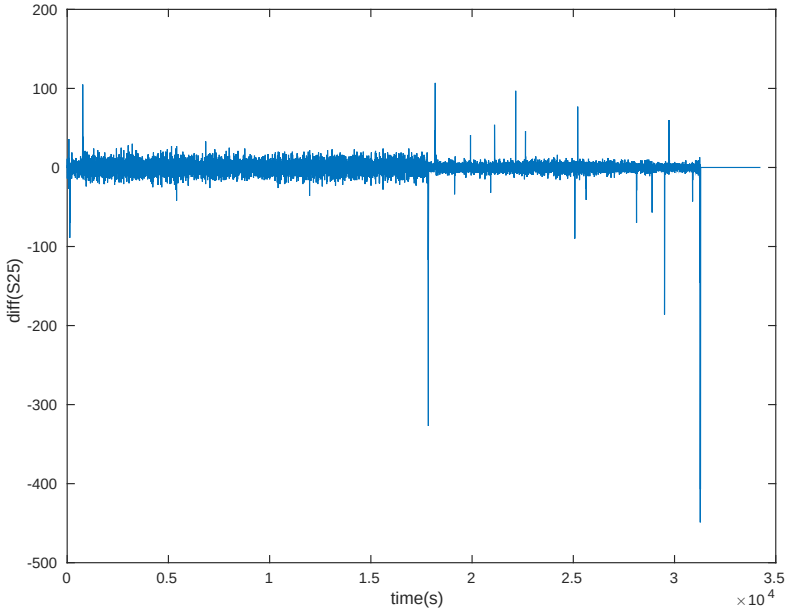


Figure 5: Difference between adjacent points for signal S_{25} for understanding the stationary in the time series.

test to measure the significance of the correlation between signals. The p-value is the probability of getting a correlation as large as the observed value when the true correlation is zero (i.e., testing the null hypothesis of no correlation). When P_{S_i,S_j} is small, then $corr(S_i, S_j)$ is significant. In our experiment, we select which have $corr(S_i, S_j) > .9999$.

With this configuration, 12 highly correlated signals are extracted. We selected the most highly correlated signals let us denote them with S_{25} and S_{24}. Figure 3 provides these two signals pictorially.

4.3.2 Model and Order Selection

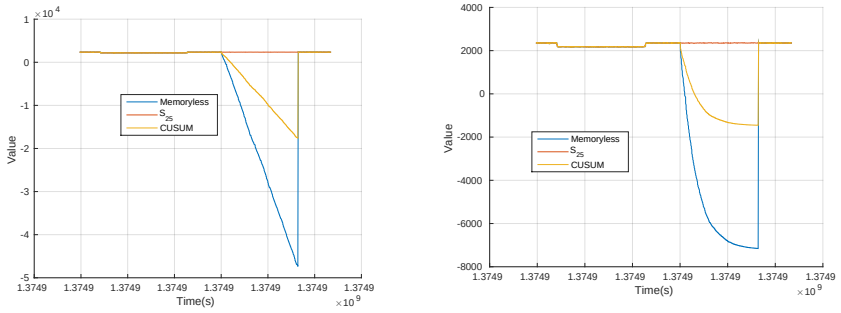
For our experiments, S_{25} is selected as main or targted signal and S_{24} is used as correlated/assistant signal of S_{25}.

For model selection, we first calculate the ACF (auto correlation function) and PACF (partial autocorrelation function) of signal S_{25}. For ACF, we do not see any decay but PACF, there is a cut off after 1 lag (please see figure 4).

Moreover, when we calculate differences (with *diff* function in MatLAB) between adjacent elements of S_{25}, we find the resulted signal is stationary (see figure 5) as the mean, variance, etc. are not changing over time. And mean is around zero (0). This leads us to select ***AR(1)***. For more details please read chapter 3 [7].

For CUSUM, we picked δ as two times mean of the residues which we find better from graphical view.

5 Results



(a) Memoryless and CUSUM with Attacks for signal S_{25} with $\tau = 100$.

(b) Memoryless and CUSUM with Attacks for signal S_{25} with $\tau = 100$ with correlated signal S_{24}

Figure 6: Attack on signal S_{25} for detection technique with memoryless and CUSUM for both single and correlated signal

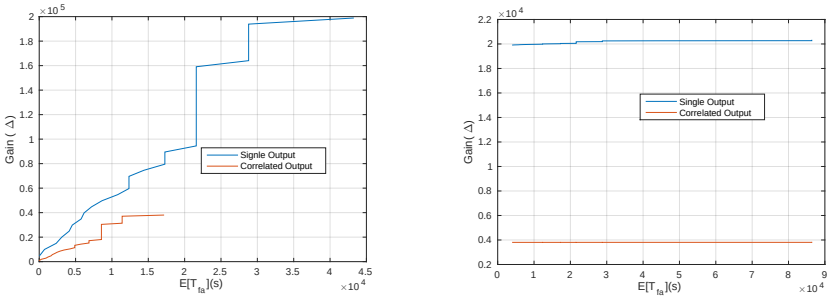
We launch an attack for same interval and tried to understand the gain (Δ) for memoryless and CUSUM techniques. We use either S_{25} or two correlated signals (S_{25} and S_{24}) for prediction single signal (S_{25}). For correlated signals, we build the model with these two signals and use the model for predicting single signal. Please see the output in figure 6a for attack and calculate gain while we use memoryless and CUSUM as detection techniques. In the same way, figure 6b shows, the deviation for attack keep undetected for memoryless or one-shot and CUSUM.

To understand more about the cost (getting a false alarm) to receive higher gain (deviation), we did experiments with a range of τs. The expected time for getting a false alarm for a specific τ and corresponding the gain are calculated. The range of τ for these experiments are started from 10 and end at 400 by increasing 10. Thus, it results 40 points. The output of these results generates figure 7b and figure 7b.

6 Discussion

From figure 6a, it is apparent, the gain with memoryless is higher (as deviation is large) than the gain with CUSUM as detection technique. If attacker knows the proper threshold, she can launch attack with undetected and receive more gain if the detection technique is memoryless.

Almost same behavior we receive when we use two correlated signals for detection. That means, gain received for memoryless one (see figure 6b) is quite significant. However, in this case, gain is considerably lower than gain with single signal (deviation is almost 7 times higher). Thus, detecting false data injection in time series, CUSUM is better than memoryless one. If we combine



(a) Gain vs expected time for a false alarm for various τs for memoryless detection method

(b) Gain vs expected time for a false alarm for various τs for CUSUM detection method

Figure 7: Gain vs. expected time for a false alarm for single vs. Correlated outputs

two or more signals, then detection approach become more stronger.

Our understanding becomes more firm, when analysis the figure 7. Figure 7a confirms that the relationship between expected time for a false alarm and gain is almost linear when we use single signal. As the time between two false alarms become larger, attacker receives more gain. In the same way, when we use correlated signals for memoryless detection technique, gain is not as much as impressive like single signal (self) usage. In later case, even if the time between two false alarms is increased, gain is not increase at all (highest gain about 40000 when expected time for false alarm is around 1.7×10^4 seconds).

However, we receive a flat relation between $E[T_{fa}]$ and Δ in case of CUSUM as a detection method. And using correlated signals is promising when we compare with technique using single signal as like we mentioned earlier. Thus, CUSUM with combined signals are best among all approaches, here we discussed. However, when we tolerate small time between two false alarms, memoryless and CUSUM with single signal may be good options (of course not practical in many cases).

7 Related Work

Our work is much related to work presented by [6]. In this work, they autoregressive modeling and control limits to understand the behavior of a time series with continuous value. However, they do not provide much details for their attacks and which are relatively straightforward. On the other hand, we consider two detection techniques based autoregressive model with single and correlated signals and analyze their strength using a new metric (maximum deviation). The motivation of new metric emerges from the inconsistency of current works. Discussion of them is out of scope of this project.

8 Future Work

In future, we plan to extending this work by increasing number of correlated signals and understand the trade off for this increment. The detection techniques and metric should be analyze with various scenarios. Another problem we like to consider, the cost of implementing this approach when it needs to deploy a detection system for a significant amount of signals and a possible architecture for it.

9 Conclusion

In this project, we evaluated two detection techniques based on auto-regression model for bad data in time series for register in PLC using our own proposed metric, maximum deviation. We followed a systematic approach for selecting the well fitted model (among various versions of autoregressive models) for the targeted data. This kind of metric will help researchers a common way to evaluate their methods. From our experiments, we come to a conclusion that when we use more than one signal (signal itself), the detection scheme becomes more strong. Beside, CUSUM like change detection approach in time series is more helpful for detecting bad data.

10 Acknowledgement

This work is an extended part of our group research project lead by Dr. Cardenas, my supervisor. So, I like to thanks for his help and guide. I also like to thanks my lab mates David Urbina (for launching attack) and Junia Valente (for signal correlation) for their kind help and sharing idea.

References

[1] Åström, K. J., and Eykhoff, P. System identification—a survey. *Automatica 7*, 2 (1971), 123–162.

[2] Box, G. E., Jenkins, G. M., and Reinsel, G. C. *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[3] Carcano, A., Coletta, A., Guglielmi, M., Masera, M., Fovino, I. N., and Trombetta, A. A multidimensional critical state analysis for detecting intrusions in scada systems. *Industrial Informatics, IEEE Transactions on 7*, 2 (2011), 179–186.

[4] Cardenas, A. A., Amin, S., Lin, Z.-S., Huang, Y.-L., Huang, C.-Y., and Sastry, S. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security* (2011), ACM, pp. 355–366.

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

[6] Hadziosmanovic, D., Sommer, R., Zambon, E., and Hartel, P. Through the eye of the PLC: Towards semantic security monitoring for industrial control systems. In *Annual Computer Security Applications Conference (ACSAC); Available as ICSI Tech-Report TR 13 003* (2014).

[7] MathWorks, I. Econometrics toolbox user guide, October 2014.

[8] Wang, Y., Xu, Z., Zhang, J., Xu, L., Wang, H., and Gu, G. Srid: State relation based intrusion detection for false data injection attacks in scada. In *Computer Security-ESORICS 2014*. Springer, 2014, pp. 401–418.

[9] Williams, T. J. The purdue enterprise reference architecture. *Computers in industry 24*, 2 (1994), 141–158.