# Detectors of Smart Grid Integrity Attacks: An Experimental Assessment

Simona Bernardi, Raúl Javierre, José Merseguer
*Departamento de Informática e Ingeniería de Sistemas*
*Universidad de Zaragoza*
Zaragoza, Spain
{simonab,758906,jmerse}@unizar.es

José Ignacio Requeno
*Dpto. de Sistemas Informáticos y Computación*
*Universidad Complutense de Madrid*
Madrid, Spain
jrequeno@ucm.es

***Abstract*— Today cyber-attacks to critical infrastructures can perform outages, economical loss, physical damage to people and the environment, among many others. In particular, the smart grid is one of the main targets. In this paper, we develop and evaluate software detectors for integrity attacks to smart meter readings. The detectors rely upon different techniques and models, such as autoregressive models, clustering, and neural networks. Our evaluation considers different "attack scenarios", then resembling the plethora of attacks found in last years. Starting from previous works in the literature, we carry out a detailed experimentation and analysis, so to identify which "detectors" best fit for each "attack scenario". Our results contradict some findings of previous works and also offer a light for choosing the techniques that can address best the attacks to smart meters.**

***Index Terms*—integrity attacks, smart grids, anomaly-based detection, ARIMA, clustering, neural networks**

## I. Introduction

Smart grids are complex cyber-physical systems, built on top of electrical power infrastructures. Although smart grids allow a more resilient, secure and reliable electricity supply, they are vulnerable to cyber-attacks [**?**]. Each type of attack focuses on a different aspect of the smart grid. But, any successful intrusion will lead to a malfunction of the power supply chain, information leaks, energy theft and many other security, safety or economical issues [**?**].

Energy thefts [**?**] and private information leaks [**?**], [**?**] are ranked as the most prominent threats to smart grids. These threats are closely linked to the advanced metering infrastructure (AMI), that provides smart meters, in consumers' houses. Smart meters allow for recording consumption, adjusting power delivery and many other features. But according to estimates, the theft of electricity, in this technical context, is as much as $96 billion every year globally [**?**]. Hence, protecting the smart grid is an economical need, but also a social one since outages may cause even more severe consequences.

This work wants to contribute to improve the technology for protecting the AMI. In particular, that part of the technology in charge of preserving the integrity of the smart meters. For this goal, we carry out a detailed evaluation of techniques, that can discover attacks to the smart meter infrastructure. Such techniques, when implemented as software, are called "integrity attack detectors". Concretely, we evaluate five different types of detectors from the literature. The evaluation is carried out considering different "attack scenarios" and datasets. Our findings allow to know which category of detectors fits better for a kind of attack. We believe that the importance of our work strives both on cutting paths that should not be followed, i.e., techniques that should not be used, at least for some kind of attacks, and on finding the paths that promise good results for protecting the AMI.

### *A. Related work*

The detection and identification of frauds started with statistical techniques [**?**], [**?**]. Since then, a variety of software approaches to detect malicious attacks in power systems appeared due to the fast development and exposition of AMI in smart grids [**?**], [**?**].

Usually, AMI energy-theft attacks are based on fault data injection or mimicking customers' consumption profiles. Electricity theft detection methods range from specialised methods on well-defined attack strategies to the discovery of general consumption behaviour anomalies [**?**]. The goal is to distinguish between normal and anomalous energy usage patterns in order to classify the samples in the dataset into one of the predefined attack profiles.

Detection methods generally fall into three categories: classification-based methods, state-based methods and game theory-based ones. Classification-based schemes stem from data mining and machine learning techniques. These methods look for patterns in the electricity consumption of a customer or a group of customers over a period of time. Classification-based techniques include, for instance, Support Vector Machine (SVM) approaches [**?**], average-based detectors [**?**], time series and autoregressive (integrated) moving average (ARIMA) detectors [**?**], neural networks [**?**], Principal Component Analysis (PCA) [**?**], relative frequency distribution-based detectors [**?**], fuzzy classifiers [**?**] or P2P computing algorithms [**?**]. In this work, we consider a subset of detectors belonging to this category.

In order to improve the accuracy and detection rates, state-based methods combine data from multiple sources (e.g., the monitoring state of physical sensors and devices) [**?**].

Finally, in game theory-based approaches [**?**], [**?**] the problem of integrity attack detection is formulated as a game between the electricity company and the attacker. In the game, the goal of the electricity thief is to steal a predefined amount of electricity while reducing the likelihood of being detected. Conversely, the utility companies want to maximize the probability of detection and decrease the operational cost of managing the anomaly detection mechanism. The game theory-based detection schemes return a reasonable, sometimes optimal, solution to minimize the electricity losses as a result of energy theft.

The paper is structured as follows. Section II settles the bases of our experimentation. Section III describes the detectors under evaluation. Section IV proposes the evaluation approach. Section V summarizes our results. Conclusions are drawn in Section VI.

## II. Basis of the evaluation

The inputs for assessing a detector will be the information it has to evaluate and the scenarios in which it will be tested. Hence, the basis for our experimentation relies on the datasets used and on the scenarios considered. In the following, we explain their ground.

### *A. Datasets*

We have used two datasets, both from the Ireland's Commission for Energy Regulation [**?**]. One of them provides information on electricity consumption of customers, the other about gas consumption. Both are available, for research purposes, upon request, and the information provided is anonymous. Both datasets are characterized by the same meta-data:

- smart meter identifier (meterID),
- timestamp (coded with a number that indicates the time of the reading), and
- the electricity/gas consumption (kWh).

The frequency of the readings is the same for both datasets, that is, every half an hour.

The electricity dataset considers 6,445 smart meters, during a period of 76 weeks, categorized as residential, small and medium enterprises (SME), and unclassified. The gas dataset considers 1,576 smart meters, during a period of 78 weeks, and it is partitioned in testing and control groups. The testing groups are categorized according to the different types of contract.

*a) Datasets preprocessing:* The information provided by the datasets was not complete for all smart meters readings. Hence, we needed to process it before our evaluation. We firstly identified the smart meters with complete readings, i.e., those having 336 readings per week. Among the valid smart meters, we randomly chose 500 from the electricity dataset and another 500 from the gas one, ensuring that the proportion was preserved according to the categorization of the original datasets. We considered that 1,000 smart meters was a good balance between computational needs and representativeness of the datasets[1]. Regarding the period, we could get 75 weeks, for all selected smart meters, in the case of the electricity dataset, i.e., only one week got lost. For the 500 gas smart meters, we achieved 74 complete weeks, i.e., four weeks got lost.

[1]Consider that the experiments lasted about 9 days, in a powerful cluster, as described in Subsection V-A.

It is worth to observe that the electricity dataset has been used in [**?**], [**?**], [**?**]. Unfortunately, their experiments are tricky to reproduce, precisely due to the absence of information about how the missing information was treated. Also, as far as we know, there are no public repositories offering the original experiments. Therefore, a fair comparison between their findings and ours will not be possible.

### *B. Scenarios*

We assume that the original information in the datasets is not affected by integrity attacks. Thus, we consider such information as a proper scenario, that we call *normal scenario*. In Figure 1, such scenario is represented by the curve with solid line. This curve depicts the electricity consumed by a smart meter, concretely #1024, every half an hour during two days. We can observe a periodic behaviour following the normality of daily life, which is characterized by off-peak and peak hours of consumption.

In addition to a *normal scenario*, we consider *attack scenarios*. The information in such scenarios is assumed to have been tampered with by external malicious agents. We get such behaviours by generating synthetic datasets from the original ones. The attack scenarios considered are the following ones.

*a) False data injection (FDI):* The goal of FDI [**?**] is to defraud the energy utility. It changes, each smart meter reading, by a given percentage $x\%$ ($x \in [0, 100]$). Hence, it achieves that the customer pays less than consumed. We have considered two FDI scenarios, cf. Figure 1a.

*b) Average attack (Avg):* It has the same goal as FDI. However, data are manipulated by replacing each reading in the week by the average consumption of the week multiplied by the realization of a uniform random variable $r \in \mathcal{U}[1 - a, 1 + a]$, where $0 < a$. This strategy [**?**] is used to go unnoticed by average-based anomaly detectors, e.g., the *Min-Avg* detector described in Section III. We have considered one attack scenario of this type, cf. Figure 1b.

*c) Random scale attack (RSA):* It can be used to achieve different goals [**?**]. If, on average, consumption data are over-reported, then the attack may cause instability in the smart-grid system. Whereas, if on average the consumption data are under-reported, then the attack causes energy theft. In particular, an RSA scenario is obtained from the normal scenario by multiplying each consumption with a realization of a uniform random variable $r \in \mathcal{U}[a, b]$, where $0 < a < b$. We have considered two RSA scenarios, one for each goal, cf. Figure 1c.

*d) Swap attack:* Also this type of attack aims at defrauding the energy utility by paying less than consumed. It assumes a time-of-use contract, where the cost of energy depends on peak and off-peak periods [**?**]. We assume the peak period from 9:00am to midnight and the off-peak period from midnight to 9:00am. Then, the attack consists in swapping the consumption data registered by a smart meter between the two periods, cf. Figure 1d.

Table I offers an initial insight about potential benefits of each kind of attack. In the case of the *normal* scenario the bill cost would be 1,104.59 euros[2]. In particular, in all the attack scenarios but the RSA[0.5,3], the attacker gain corresponds to the percentage of money saved by the customer compared to the bill cost of the normal scenario. Whereas in RSA[0.5,3], that aims at over-reporting the consumption, indicates the percentage of money overpaid by the customer to the utility.

| **Attack scenario** | **Bill cost (€)** | **Attacker gain** (%) |
|---|---|---|
| FDI_10 | 110.46 | 90% |
| FDI_30 | 331.38 | 70% |
| Avg[0.5,1.5] | 982.53 | 11% |
| RSA[0.25,1.1] | 749.40 | 32% |
| Swap | 744.58 | 33% |
| RSA[0.5,3] | 1,944.22 | 76% |

TABLE I: Consequence of successful attacks to smart meter #1014, over a period of 60 weeks

## III. Attack detectors

Detectors are pieces of software, that analyse a set of data to try to foresee anomalies. From the analyses of the consumption data, our detectors try to predict attacks on integrity. Basically, they decide whether the data belong or not to a *normal scenario*. We have implemented the detectors using Python [?].

Table II presents the detectors we implemented for assessment. Most of them belong to the literature, the others are variants that we propose, i.e., ARIMAX and JSD. In the following we provide a brief description of the detectors.

| **Name** | **Technique** |
|---|---|
| Min-Avg | Comparison of the minimum of the average consumptions in the past |
| ARIMA, ARIMAX | Auto-Regressive models |
| PCA-DBSCAN | Principal component analysis, clustering |
| KLD, JSD | Comparison of relative frequency distributions of consumption |
| NN | Deep learning |

TABLE II: Detectors considered in the experiments

*a) Minimum average detector:* The *Min-Avg* detector was proposed in [?] with a straightforward design. From a time series of consumption data, representing a training period of $n$ weeks, the prediction model calculates the average consumption of each week and gets the minimum, $m$. Then labelling as "anomalous", those weeks whose average consumption is less than $m$.

*b) Auto-Regressive models: ARIMA* (Auto-Regressive Integrated Moving Average) was proposed by [?]. *ARIMAX* (Auto-Regressive Integrated Moving Average with Explanatory Variable) is a variant that, unlike ARIMA, includes exogenous Fourier variables to take into account the daily frequency of consumption. Both detectors construct an auto-regressive model, from time series of consumption data.

An ARIMA(X) prediction model is characterized by a triplet $(p, q, d)$ [?]:

- $p$ is the autoregression order,
- $q$ is the moving average order, and
- $d$ is the degree of differentiation of the time series.

The more accurate the parameters are, the better the model fits the time series.

We use the Python API `auto_arima`[3] to set the parameters automatically. Finally, we obtain an optimal prediction model, with respect to the BIC (Bayesian Information Criterion). BIC is generally preferred for large samples [?], as it is the case of our data sets.

Once we get the ARIMA(X) model, it is used to predict the consumption every half hour, with a confidence interval of 95%. The detection criterion verifies whether the registered consumption, at timestamp $t$, lies within the confidence interval of the prediction at $t$. If not, the consumption will be labeled as "anomalous".

*c) PCA and clustering:* The *PCA-DBSCAN* detector was proposed by [?]. It combines two data-mining techniques:

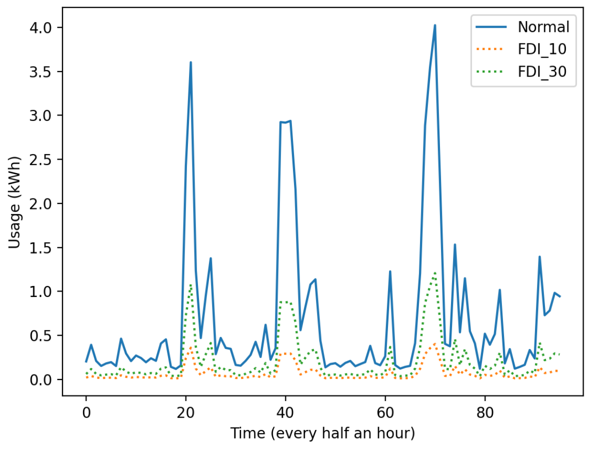
- First step. It uses PCA (Principal Component Analysis) [?] to reduce the dimensionality of the consumption data. Unlike the other detectors, that build a prediction model per smart meter, it uses the entire training dataset. So, it applies PCA to the consumption data registered by all the meters of the smart-grid. The result is a reduced matrix, that characterizes each training week with a point in a two-dimensional space.
- Second step. It applies DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [?] to the set of points, of the reduced matrix, that correspond to a given smart meter. Then, it creates a prediction model as a cluster of points, that represents the *normal* behaviour. The points outside the cluster will be labelled as "anomalous".

DBSCAN uses two parameters to create the model. The radius of the circular region around a point (*eps*), and the minimum number of points at a Euclidean distance less than *eps* point. We set these parameters using the Rousseeuw and Croux estimator [?] for the calculus of the radio, and simple majority for the second parameter, as in [?]. It is worth to observe that the condition may not be satisfied for any of the points in the space, which means that no grouping is generated. In this case the detector cannot be applied.

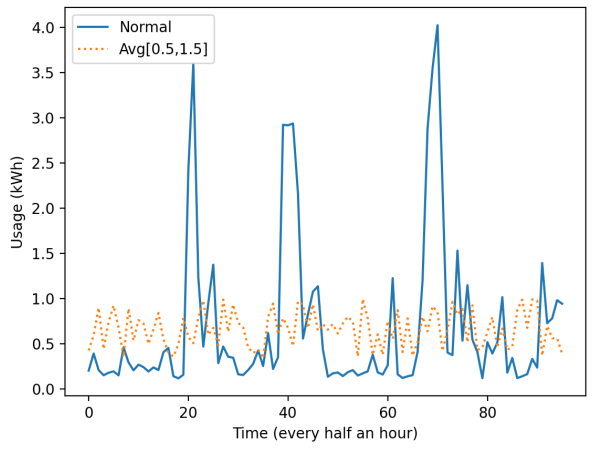
*d) Comparison relative frequency distributions:* Kullback-Leibler Divergence (KLD) was proposed by [?]. We have also implemented a variant, using Jensen-Shannon Divergence (JSD). Both detectors classify a week, as

[2]The reference used to calculate the cost of the energy was https://selectra.es.

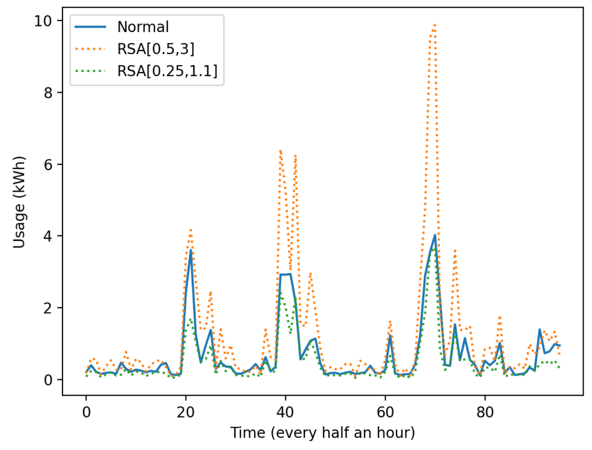
[3]Python Library `pmdarima`: http://alkaline-ml.com/pmdarima.



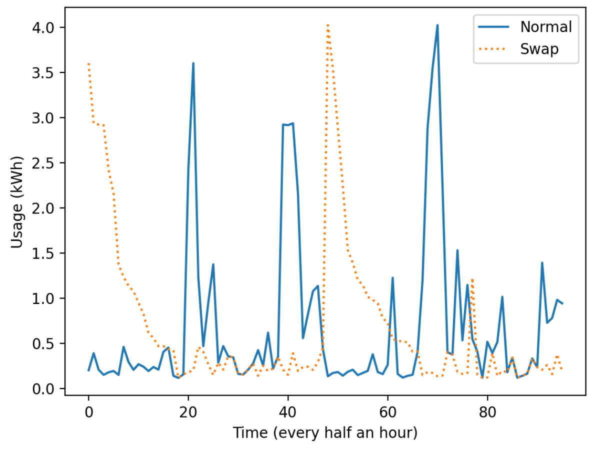
(a) False Data Injection attacks



(b) Average attacks



(c) Random scale attacks



(d) Swap attack

Fig. 1: Consumption in the *normal* scenario vs. consumption in *attack* scenarios (electricity dataset, smart meter #1014 over two days of observations)

anomalous, when its distribution does not match the distribution of the training data. The detectors differ in the function used to quantify such divergence [**?**].

The prediction model is built based on the divergence function and the relative frequency distributions associated to the training weeks. The latter are characterized by the same frequency intervals, in particular we considered 30 contiguous intervals of equal length between the minimum and maximum consumption of the training set. The divergence function calculates the deviation between the distribution of a week $A$ and the distribution of all $n$ weeks of training, such $n$ values define the prediction model. The detection criterion relies on the 95 percentile threshold of the $n$ values. If the divergence of the distributions is greater than the threshold, the week is labelled as “anomalous”.

*e) Deep learning:* In [**?**] was proposed a detector based on neural networks (*NN*). It relies on deep learning. Unlike previous detectors, *NN* follows a supervised learning approach and it is able to identify the type of attack. Thus, it requires to train different scenarios, including the normal and attack scenarios described in Subsection II-B.

The prediction model is a neural network made of two layers. The first layer has 10.000 neurons. The second one has $N$ neurons, where $N$ is the number of scenarios. The neural network is trained for each scenario, which consists of $n$ training weeks partitioned into samples of 48 hours, i.e., two days. We have considered several statistical qualifiers to characterize each sample within a scenario. In particular, mean, standard deviation, quartiles, interquartile range, and the difference between the last and the first sample.

From a testing sample, the *NN* produces a probability distribution over the set of scenarios. The sample is classified according to the scenario with the highest probability. Thus, in case of attack scenario, the sample is labelled as “anomalous”.

## IV. Evaluation approach

Let $\mathcal{E} = \langle \mathcal{M}, \mathcal{D}, \mathcal{S} \rangle$ be an experimentation plan, where $\mathcal{M}$ is a set of smart meters, $\mathcal{D}$ a set of detectors to be assessed and $\mathcal{S}$ a set of scenarios. In an $\mathcal{E}$ plan, for each triplet $(m, d, s) \in \mathcal{M} \times \mathcal{D} \times \mathcal{S}$, we collect a set of metrics. Then, we distinguish between *basic metrics* and *aggregated metrics*.

*Basic metrics* are computed for each experiment $(m, d, s)$. They are summarized in Table III[4], as follows:

- Partial confusion matrices $q_{mds}$ of each experiment $(m, d, s)$, that is matrices with either the true positives/false negatives column or the false positives/true negatives column.
- Performance metrics: $tb_{mds}$ as the execution time to build a prediction model for an experiment $(m, d, s)$, and $tp_{mds}$ as the execution time to make predictions for $(m, d, s)$.

| | |
|---|---|
| $q \equiv \begin{bmatrix} tp & fp \\ fn & tn \end{bmatrix}$ | $\begin{bmatrix} \text{true positives} & \text{false positives} \\ \text{false negatives} & \text{true negatives} \end{bmatrix}$ |
| $\langle tb, tp \rangle$ | ⟨time to build the model, time to get predictions from the model⟩ |

TABLE III: Basic metrics

From the basic metrics, we compute *aggregated metrics of quality*, as shown in Table IV. They are obtained as statistics values from the basic metrics.

| | | |
|---|---|---|
| $tpr$ | true positive rate | $tp/(tp + fn)$ |
| $tnr$ | true negative rate | $tn/(fp + tn)$ |
| $prev$ | prevalence | $(tp + fn)/(tp + fn + fp + tn)$ |
| $ba$ | balanced accuracy | $(tpr + tnr)/2$ |
| $mcc$ | Matthews correlation coefficient | $\frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$ |

TABLE IV: Aggregated metrics of quality

The *aggregated metrics of quality* are proposed at two levels. In particular, the single scenario level and the global level. The latter to provide an insight of the quality of the detectors for all the scenarios.

At single scenario level $s \in \mathcal{S}$, we are interested in the sensitivity, in case of attack scenarios, and the specificity, in case of the normal scenario. Indeed, the former (also called true positive rate, $tpr$, or recall) indicates how often a detector $d$ correctly predicts an actual attack. The latter (also called true negative rate, $tnr$) indicates how often a detector $d$ correctly predicts that there is not an attack. Such metrics are computed from the partial confusion matrices (i.e., matrices characterized by one zero-column) $q_{ds} = \sum_{m \in \mathcal{M}} q_{mds}$, where the sum is the matrix addition operator.

At the global level, for each detector $d \in \mathcal{D}$, we are interested in computing not only aggregated metrics of quality, but also performance metrics.

[4]We omit the subindices in the tables to avoid burdening.

Concerning aggregated metrics of quality, besides the sensitivity and specificity, we have considered the balanced accuracy and the Matthews correlation coefficient. Both metrics are useful in case of unbalanced data [**?**], as it occurs in our experiments, where there are six types of attack scenarios but only one normal scenario. Moreover, the prevalence metric indicates how often the attack condition actually occurs in the overall observed results. All quality metrics are computed from the full confusion matrices $q_d = \sum_{s \in \mathcal{S}} q_{ds}$.

Finally, concerning performance metrics, we have considered those already given, i.e., $tb$ and $tp$. But now, they are also aggregated, that is computed as means over all smart meters $m \in \mathcal{M}$ and all scenarios $s \in \mathcal{S}$.

## V. Experimentation results

This section presents the experimentation plans and the results obtained.

### A. Experimentation set-up

We set up two experimentation plans, $\mathcal{E}_{el}$ and $\mathcal{E}_{gas}$, using the datasets described in Subsection II-A. Each plan considers 500 smart meters, with complete readings, and all scenarios in Subsection II-B. $\mathcal{E}_{el}$ uses all detectors in Section III, whereas $\mathcal{E}_{gas}$ could not apply PCA-DBSCAN, next subsection will detail the reason.

Algorithm 1 describes the experimentation launcher, at a high-level of abstraction. It takes $\mathcal{E}$, as input, and produces basic metrics, as described in Section IV. Concretely, the confusion matrices $\mathcal{Q}$, and execution times $\mathcal{P}$.

**Algorithm 1:** Experimentation launcher

```
Data: $\mathcal{E} = \langle \mathcal{M}, \mathcal{D}, \mathcal{S} \rangle$ (smart meters, detectors, scenarios)
Result: $\mathcal{Q}$ (quality metrics), $\mathcal{P}$ (performance metrics)
1  $\mathcal{Q} = \emptyset$, $\mathcal{P} = \emptyset$;
2  foreach $m \in \mathcal{M}$ do
3      foreach $d \in \mathcal{D}$ do
4          $train$ = getTrainingDataset($d, m, dsPath$);
5          $\langle \mu, tb \rangle$ = buildModel($d, train$);
6          foreach $s \in \mathcal{S}$ do
7              $test$ = getTestingDataset($s, m, dsPath$);
8              $\langle pred, obs, tp \rangle$ = predict($d, test, \mu$);
9              $q$ = getConfusionMatrix($pred, obs, s$);
10             $\mathcal{Q} = \mathcal{Q} \cup \{q\}$;
11             $\mathcal{P} = \mathcal{P} \cup \{\langle tb, tp \rangle\}$;
12         end
13     end
14 end
```

We partitioned the consumption readings in two sets, *training* and *testing*. $\mathcal{E}_{el}$ and $\mathcal{E}_{gas}$ used the same percentage of readings, for all the smart meters. Concretely, 80% for training and 20% for testing, over all the observed period. The *training* set is loaded from secondary storage (line 4). It depends on both, the smart meter $m$ and the detector $d$, and it is used to build the prediction model $\mu$ (line 5). Besides, the execution time $tb$, required to build the model, is monitored. Then, for each scenario $s$, the corresponding *testing* set, associated to $m$, is loaded (line 7), and predictions of $d$ are made using the model $\mu$ (line 8). As a result of the predictions, in addition to the execution time $tp$, we get the number of attacks predicted, $pred$, and the number of observations, $obs$. The latter are used to compute the confusion matrix (line 9).

As experimental environment, we used a cluster with 8 physical cores, Intel Xeon Gold 6254 CPU, 32GB RAM, 1 TB SSD, running Ubuntu 20.04 OS. The time needed for the experimentation plans to complete took nine days, approximately. We carried out them twice, so to ensure results consistency.

## B. Results

This section presents the results obtained for the quality and performance metrics, at both levels, scenario and global.

First, we provide an insight of the global metrics, for both plans. We observe that, $\mathcal{E}_{el}$ and $\mathcal{E}_{gas}$ are characterized by the same prevalence ($prev = 0.92$). Hence, resulting in unbalanced columns of the confusion matrices, $q_d$. This confirms that the total number of attack events is dominant, with respect to the normal consumption.

All the detectors managed to make predictions for the entire set of smart meters, but PCA-DBSCAN. Indeed, PCA-DBSCAN only succeeded for 66.2% of smart meters, in case of $\mathcal{E}_{el}$. Whereas in $\mathcal{E}_{gas}$ it could not be applied[5].

| **Detector** | **Quality metrics** | | | | **Perf. metrics** | |
|---|---|---|---|---|---|---|
| | *tpr* | *tnr* | *ba* | *mcc* | *tb* (s.) | *tp* (s.) |
| Min-Avg | 0.441 | 0.978 | 0.709 | 0.236 | 0.086 | 0.020 |
| ARIMA | 0.040 | 0.934 | 0.487 | -0.035 | 67.499 | 0.134 |
| ARIMAX | 0.068 | 0.928 | 0.498 | -0.005 | 974.368 | 0.152 |
| PCA-DBSCAN | 0.145 | 0.954 | 0.550 | 0.080 | 4.326 | 5.929 |
| KLD | 0.646 | 0.834 | 0.740 | 0.271 | 0.008 | 0.002 |
| JSD | 0.628 | 0.867 | 0.747 | 0.278 | 0.009 | 0.002 |
| NN | 0.932 | 0.137 | 0.535 | 0.073 | 9.078 | 1.654 |

TABLE V: Global metrics results for $\mathcal{E}_{el}$

*1) Global level results:* Table V shows the global metrics results of $\mathcal{E}_{el}$. It discloses that the sensitivity ($tpr$) is very low for Min-Avg, ARIMA(X) and PCA-DBSCAN, medium for KLD and JSD, while the NN detector is highly sensitive to attacks. On the other hand, with respect to the specificity ($tnr$), the detectors behave in opposite manner. In particular, the NN detector raises false positives for most of the normal events.

The balanced accuracy ($ba$) and the Matthews correlation coefficient ($mcc$) provide reliable statistical rates in the evaluation of the overall quality of the detectors. The first reveals that the Min-Avg, KLD and JSD are fair detectors, yielding a correct prediction between $70.9\% - 74.7\%$ of the sample experiment, while the remaining detectors behave quite badly. The $mcc$ metric, whose domain is $[-1, +1]$, confirms this interpretation. We observe that best $mcc$ values correspond to Min-Avg, KLD and JSD, although they are still close to zero. The rest of the values are approximately equal to zero, thus indicating that the detectors are no better than a random flip of a fair coin.

Concerning mean execution times, ARIMA(X) detectors have definitely the poorest performance, in building the prediction models, than the rest (crf. $tb$ values). Moreover, ARIMAX is about 14 times slower than ARIMA, however the quality improvement, of the prediction model, with respect to the latter is negligible (crf. $ba$ values). The Min-Avg, KLD and JSD detectors are very fast (order of milliseconds), both for building the model and for making predictions. PCA-DBSCAN is the slowest detector in making prediction, the main reason is that we considered the second phase (DBSCAN) as part of the prediction step.

Results of $\mathcal{E}_{gas}$ are similar to those in $\mathcal{E}_{el}$. Table VI discloses that, in general, quality metrics are slightly worse, for all detectors. The only exception is the specificity of KLD and JSD, that increases of more than 0.1.

| **Detector** | **Quality metrics** | | | | **Perf. metrics** | |
|---|---|---|---|---|---|---|
| | *tpr* | *tnr* | *ba* | *mcc* | *tb* (s.) | *tp* (s.) |
| Min-Avg | 0.144 | 0.991 | 0.568 | 0.110 | 0.086 | 0.023 |
| ARIMA | 0.026 | 0.934 | 0.480 | -0.066 | 50.562 | 0.078 |
| ARIMAX | 0.035 | 0.936 | 0.485 | -0.043 | 630.984 | 0.155 |
| PCA-DBSCAN | | | | | | |
| KLD | 0.477 | 0.965 | 0.721 | 0.246 | 0.008 | 0.002 |
| JSD | 0.512 | 0.971 | 0.742 | 0.268 | 0.009 | 0.002 |
| NN | 0.939 | 0.113 | 0.526 | 0.057 | 9.021 | 1.668 |

TABLE VI: Global metrics results for $\mathcal{E}_{gas}$

*2) Scenario level results:* Table VII shows the results for both experimenation plans. The metrics are computed from the partial confusion matrices $q_{ds}$. The sensitivity at this level provides a useful feedback about the detection capabilities for the different types of attacks. The specificity has been already discussed (cfr. columns $tnr$ of Tables V and VI).

Let us consider first $\mathcal{E}_{el}$ (upper part). Min-Avg detector, as expected, is able to detect deterministic attacks (i.e., FDI), but not the others. ARIMA(X) and PCA-DBSCAN are very poor detectors, for all types of attacks. KLD and JSD are good to detect FDI and Avg attacks, however they are not able to recognize Swap attacks nor RSA, that under-reports consumptions. Moreover, they behave different with respect to the RSA that over-reports consumptions. In particular, KLD performs better than JSD. Finally, NN is excellent to detect all types of attack scenarios, however its main problem is the high number of false positives (i.e., very low specificity).

The results for $\mathcal{E}_{gas}$ (lower part) confirm the previous observations on the global metrics for the same dataset. Concerning the sensitivity values, they are in general worse than the

[5]In $\mathcal{E}_{el}$, PC-DBSCAN could not generate a cluster for the remaining 33.8% of smart meters, therefore the detection criterion was not performed. In $\mathcal{E}_{gas}$, the smart meters are not highly correlated, thus, the reduction with PCA (i.e., the first step to get the prediction model) to a two dimensional space was not applicable.

Electricity experimental plan $\mathcal{E}_{el}$

| Metric | Scenario | Min-Avg | ARIMA | ARIMAX | PCA-DBSCAN | KLD | JSD | NN |
|---|---|---|---|---|---|---|---|---|
| $tnr$ | Normal | 0.978 | 0.934 | 0.928 | 0.954 | 0.834 | 0.867 | 0.137 |
| $tpr$ | FDI_10 | 0.955 | 0.002 | 0.029 | 0.281 | 0.887 | 0.909 | 0.863 |
| | FDI_30 | 0.757 | 0.002 | 0.015 | 0.187 | 0.738 | 0.789 | 0.863 |
| | Avg | 0.022 | 0.010 | 0.028 | 0.044 | 0.902 | 0.912 | 1.000 |
| | RSA[0.25,1.1] | 0.226 | 0.030 | 0.037 | 0.090 | 0.202 | 0.203 | 0.942 |
| | RSA[0.5,3] | 0.003 | 0.170 | 0.185 | 0.015 | 0.765 | 0.554 | 0.952 |
| | Swap | 0.022 | 0.066 | 0.130 | 0.051 | 0.166 | 0.133 | 0.988 |

Gas experimental plan $\mathcal{E}_{gas}$

| Metric | Scenario | Min-Avg | ARIMA | ARIMAX | PCA-DBSCAN | KLD | JSD | NN |
|---|---|---|---|---|---|---|---|---|
| $tnr$ | Normal | 0.991 | 0.934 | 0.936 | | 0.965 | 0.971 | 0.113 |
| $tpr$ | FDI_10 | 0.286 | 0.000 | 0.005 | | 0.650 | 0.807 | 0.887 |
| | FDI_30 | 0.093 | 0.000 | 0.003 | | 0.588 | 0.661 | 0.887 |
| | Avg | 0.009 | 0.002 | 0.007 | | 0.852 | 0.905 | 1.000 |
| | RSA[0.25,1.1] | 0.022 | 0.028 | 0.029 | | 0.089 | 0.047 | 0.930 |
| | RSA[0.5,3] | 0.004 | 0.120 | 0.122 | | 0.355 | 0.049 | 0.933 |
| | Swap | 0.009 | 0.066 | 0.125 | | 0.035 | 0.029 | 0.991 |

TABLE VII: Scenario level metrics (specificity, for the normal scenario, and sensitivity, for the attack scenarios).

electricity ones (the only exception is NN, which performs slightly better). In particular, we can remark that Min-Avg loses its FDI detection capabilities.

*3) Fine-grain analysis:* Figure 2 shows fine grained results for the 30% false data injection scenario (FDI_30). In particular, the boxplots show the distributions of the sensitivity values over the entire set of smart meters $\mathcal{M}$: the orange lines indicate the median values, the top and bottom sides of the boxes represent the third and first quartiles, respectively, and the vertical lines that extend from either side of the boxes represent the ranges for the bottom 25% and the top 25% of the values, excluding outliers (white dots). The Min-Avg boxplot has several outliers, nevertheless, for the 75% of the smart meters the sensitivity is approximately zero, and for the rest the sensitivity is below 0.2. ARIMA(X) definitely are not able to recognize the attack. KLD and JSD show better results than Min-Avg, but still poor. They are characterized by a high variability of the sensitivity values among the different smart meters. NN has very good detection capabilities, however we can observe many outliers.

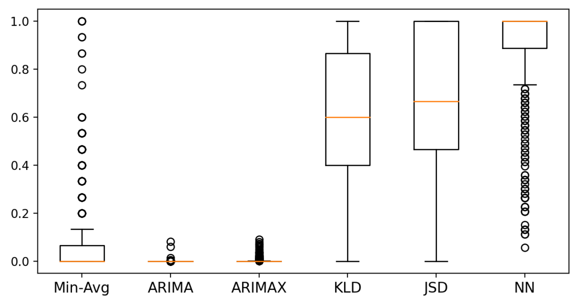


Fig. 2: Fine-grained results of the sensitivity ($tpr$) in the FDI_30 scenario (gas experimental plan).

## VI. Conclusion

This work has carried out an exhaustive experimentation, that allows us to get some interesting results. First, we have to say that a fair comparison with results of previous works has not been possible. It is true that we have borrowed techniques from the literature, although some improvements are originally ours. However, for most of the similar works it has not been possible to reproduce their experiments. This is mainly due to two problems: the not complete specification of the information used and the unavailability of the software detectors. As a result, our findings, which we completely trust, are for certain detectors different of those reported in the literature. We make available the software developed for implementing and assessing the detectors as well as the experimentation results[6].

As a synthesis of our results, we can conclude that none of the detectors is good enough to predict all types of attacks, while maintaining a low rate of false positives. On the one side, KLD and JSD perform better than the others, globally. On the other hand, NN is very promising in detecting attacks, however it needs to be adjusted so that it can discover normal behaviours as well. Nevertheless, we recognize that more experiments are needed, in order to consider different baseline periods for detection, for example, of swap attacks (e.g., 12 hours instead of week).

We have learned that, in this context, experimentation is tricky and costly. But, at the same time, more of it is needed: new detectors need to be implemented, more datasets and scenarios have to be considered, also different plan settings will offer more insights. Our intention is to completely automate the experimental process. Therefore, we are currently working on a software framework, that allows researchers to plug new detectors, scenarios and datasets.

[6]URL: https://github.com/DiasporeUnizar/smartest.

## Acknowledgment

This work has been supported by the project MEDRESE-RTI2018-098543-B-I00, funded by the Spanish Ministry of Science, Innovation and Universities.