Robust Anomaly based Attack Detection in Smart Grids under Data Poisoning Attacks

Shameek Bhattacharjee
Western Michigan University
Kalamazoo, MI, USA
shameek.bhattacharjee@wmich.edu

Mohammad Jaminur Islam
Western Michigan University
Kalamazoo, MI, USA
mohammadjaminur.islam@wmich.edu

Sahar Abedzadeh
Western Michigan University
Kalamazoo, , MI, USA
sahar.abedzadeh@wmich.edu

ABSTRACT

Anomaly-based attack detection methods are often used to detect data integrity or data falsification attacks in advanced metering infrastructure (AMI) of smart grids. However, there is a lack of studies on the effect of data poisoning attacks against the anomaly based attack detectors that depend on some form of machine learning. In this paper, we introduce some data poisoning attack strategies against anomaly-based attack detectors in smart metering infrastructure and show its impact. Specifically, we propose a whitebox and black box approach to poisoning attacks. Then, we propose modifications to improve the robustness of previous anomaly detection algorithms by modifying certain design choices for learning the thresholds for the anomaly detector. Specifically, we offer theoretical insights and experimental proof to explain why and when they mitigate data poisoning. These design choices include both the regression type and the loss function choice. We measure attack mitigation performance with two NIST specified metrics for CPS systems in the test set using a real smart metering dataset. Finally, we offer recommendations on energy utility's best anomaly detector design choices under varying attack parameters.

CCS CONCEPTS

• **Mathematics of computing** → **Regression analysis**; • **Computing methodologies** → **Machine learning**; • **Security and privacy**; • **Hardware** → **Smart grid**;

KEYWORDS

Anomaly Detection; Smart Grid; Data Poisoning Attack; Adversarial Machine Learning; Time Series; Interpretable ML based Security

ACM Reference Format:
Shameek Bhattacharjee, Mohammad Jaminur Islam, and Sahar Abedzadeh. 2022. Robust Anomaly based Attack Detection in Smart Grids under Data Poisoning Attacks. In *Proceedings of the 8th ACM Cyber-Physical System Security Workshop (CPSS '22), May 30, 2022, Nagasaki, Japan.* ACM, , USA, 12 pages. https://doi.org/10.1145/3494107.3522778

CPSS '22, May 30, 2022, Nagasaki, Japan

ACM ISBN 978-1-4503-9176-4/22/05…$15.00
https://doi.org/10.1145/3494107.3522778

1 INTRODUCTION

Data falsification attacks have been widely studied in the Internet of Things (IoT) telemetry and Cyber Physical System (CPS) domains. The objective of data falsification attacks is to disrupt the operational accuracy and functioning of an IoT/CPS application. To detect such attacks, artificial intelligence (AI), Machine Learning (ML), and statistics based anomaly detection approaches have been developed [1, 5, 11, 12].

In contrast, the discipline of adversarial ML deals with attacks that are analogous to data falsification attacks but designed specifically to exploit weaknesses in an ML/AI-based framework [7, 14]. In adversarial ML, attacks are broadly categorized as Evasion and Poisoning. Adversarial attacks launched during the test/deployment phase are called *evasion attacks*, while adversarial attacks launched during the training phase are known as *poisoning attacks*. Evasion attacks prevent correction classification of a specific input during the testing phase. In contrast, the poisoning attacks perturbs inputs during the training phase which prevents accurate parameter learning. As a result, the framework learns a wrong model that negatively affects the intended performance of the ML/AI framework.

To design an adversarial attack, an adversary may have exact knowledge of the ML technique and data points (White Box Approach); or may not know the model and/or all datapoints, but can observe outputs from an algorithm (Black Box Approach). The White Box approach specifies the worst-case possibility which applies to insider attacks or by an advanced persistent threat actor that gains system privileges to obtain full access to a utility's cyberinfrastructure. Such the mode of entry for full system control was used in the Ukraine Power Grid Attack [20].

There have been enough theoretical advances in evaluating vulnerabilities of ML models against adversarial attacks that are applied in the context of computer vision and image processing algorithms [7, 9]. Additionally, there exist practical works on adversarial attacks that target spam email detection [14] and various theoretical studies on attacks and defenses [16]. However, there is a lack of studies on adversarial learning in the context of smart grids. There is one work on adversarial machine learning which discusses smart grid but in the context of differential privacy [13]. For poisoning attack and defense in a general regression setting, [16] proposes an adversarial poisoning technique to craft extra points into an existing dataset to bias the prediction in a classical linear regression problem. Such an approach works in an open setting, but does not work in a smart grid since all meters need to physically registered to a utility. Some works [17] have offered game theoretic defenses for theoretical regression problems for an rational adversary.

Motivation Most of the existing work in AMI domain focuses on strategies to disrupt operations. However, there is a lack of works that specifically focus on adversarial poisoning attacks that target the machine learning aspect of anomaly based data integrity attack detectors. Therefore, we think there is a need for focusing on strategies that target ML approaches for data integrity attack detection in AMI. Furthermore, we could not find works that establish the effects of *poisoning attacks* on AMIs' data integrity protection or quantify the resilience of existing frameworks under such poisoning attacks. Finally, we could not find appropriate mitigation approaches under poisoning attacks in an AMI setting.

Dissecting an ML-based anomaly detection approach into different aspects (*regression type choice, loss function choice, threshold learning)* can help us analyze the explainability of a vulnerability and a mitigation countermeasure. In this work, we were motivated to examine how these *aspects* affect performance degradation under data poisoning attacks. Thereafter, we analyze how lessons from robust statistics and robust optimization can be applied in the context of AMI to offer mitigation and improve the dependability of the anomaly detector against such attacks. In [16], the attack in a regression setting was done by adding extra points to the training set and defense involved limiting the inclusion of these extra points in the learning. However, in our case time is a invariant design feature, so adding extra points for poisoning is not feasible.

Contributions First, we devise a fast gradient value inspired adversarial data poisoning (FGAV) attack that perturbs inputs to the threshold learning of the best performing anomaly based detector in AMI under a white-box assumption.

Second, we implement a more plausible smart meter level poisoning (RSL) which does not assume exact knowledge of the anomaly detector or model information. Third, we compare the effectiveness of FGAV attack and RSL attack under a previous anomaly-based data integrity attack detector using two metrics: (a) Impact of Undetected Attack (b) Expected Time between two false alarms.

Fourth, we propose modifications for robust learning of anomaly detection thresholds by integrating weighted regression with robust loss functions into a single approach, and show mitigation under both FGAV and RSL based poisoning attacks.

Finally, we provide a insight on the security performance against poisoning attacks with respect to (i) the effect of regression type (ii) the effect of loss function choice under varying attack variables. In particular, we found that the Cauchy-Lorentzian Loss function performs better than the more well-known Huber Loss, Mean Square Error Loss (L2), Mean Absolute Error Loss (L1) and that a weighted quantile regression is better in false alarm performance.

2 BACKGROUND AND PRELIMINARIES

The AMI consists of the smart meters deployed on houses that collect power consumption data periodically. For any hour t, and for the i-th house, the power consumption reported by the i-th smart meter is denoted as $P^i(t)$. Now we discuss the details of the anomaly detection.

2.1 Anomaly Detection Problem Specification

The model being poisoned is a time series anomaly detection method proposed in a series of previous works [1, 2, 5]. We chose this time series anomaly detection technique due to the following reasons: (1) most recent theoretical work in time series microgrid level anomaly detection technique that can detect very low data falsification margins without sacrificing false alarms; (2) the method has been shown to generalize to other CPS/IoT domains such as Phasor Measurement Units [4] and Smart Transportation Systems [6]; (3) follows the latest National Institute of Standards and Technology (NIST) specified metrics [12] (such as impact of undetected attacks and expected time between false alarms) for evaluation; (4) While many anomaly detection methods in industrial CPS, use the idea of pairwise comparison of positive covariance structure [11] for invariant design, the pairwise co-variance calculations are expensive for large community-scale smart living CPS (such as AMI). The method in [1] proposed a lightweight equivalent of the same principle. Furthermore, the chosen framework's metric shows unique deviation signatures under different data integrity attack types, that help reconstruct attack type, attack strategy, and attack severity [2].

The goal of the anomaly detection framework was to detect an incidence of orchestrated data falsification and omission attacks in a smart grid advanced metering infrastructure. In contrast, the goal of the poisoning attacks is usually to alter the training data to degrade the performance of the anomaly detection model in two ways: (1) *Integrity Violation* that increases missed detection of attacks (2) *Availability Violation* that increases false alarm rate of the anomaly detector. In this paper, we only investigate integrity violation as the goal of the poisoning attack.

While our poisoning attack is shown against a particular technique proposed in [1], it generalizes to other time series anomaly detection methods which utilize some form of regression-based learning to obtain the anomaly detection thresholds. This generality is apparent from the shared design similarities among various time series-based anomaly detection approaches.

To relate the contribution in a generic way, and yet experience the practical effectiveness of a specific framework and a specific application, we describe the anomaly detection technique [1], under the umbrella of the following overarching aspects: (1) Anomaly detection metric (2) Residuals of a Time Series (3) Learning of Thresholds from Residuals (4) Anomaly Detection Criterion.

2.2 Time Series Anomaly Detection Method

2.2.1 Anomaly detection Metric. The main anomaly detection metric is the time series of harmonic to the arithmetic mean ratio from multiple smart meters (proposed in [1, 2, 4, 6]). For achieving in-variance in the time series of the metric under benign conditions, the ratio is calculated over a strategic spatial and temporal granularity under Box-Cox transformations such that positive co-variance in a micro-grid of individual meters are maximized. Mathematically, the base invariant metric is defined as:

$$Q^r(T) = \frac{\sum_{t=1}^{W} HM(t)}{\sum_{t=1}^{W} AM(t)} \tag{1}$$

where $HM(t)$ and $AM(t)$ are harmonic and arithmetic mean of power consumption $P^i(t)$ from all smart meters in a micro-grid at time slot t with a box cox transformation. Each spatial cluster maintains the above ratio metric. The temporal granularity is indexed by a time window T of length W time slots (hours).

In the case of AMI, the previous work found $W = 24$ hours and the granularity included the whole solar village of 200 houses due to its small size. Hence for a typical year, the range of $T \in \{1, 365\}$. The same method runs in a decentralized manner for each spatial cluster. Hence, for the purpose of this paper, there is only one spatial cluster i.e., a small solar village forming a micro-grid of 200 houses.

Table 1: Table of Notations

Notation	Meaning
T	Time Window
τ	candidate threshold parameter
τ_{max}	upper model output threshold
τ_{min}	lower model output threshold
t	time slot (smallest time granularity)
ϵ	Perturbation Margin
F	Time Frame
α	No. of Inputs Perturbed
η_R	No. of Residual Points

2.2.2 Safe Margins. Safe margins quantify an expected upper and lower bound of the anomaly detection metric at any time window T. Let the expected value of the ratio metric at the T-th time window be $\mu^{Q^r}(T)$. The $\mu^{Q^r}(T)$ is defined as the cumulative weighted time average of ratios observed on the T-th window across multiple years (non-parametric statistic). In contrast, let σ_r denote the standard deviation of the probability distribution of $Q^r(T)$ samples in the training set (parametric statistic). The upper and lower safe margin is a neighborhood around the expected value $\mu^{Q^r}(T)$, that is controlled by a scalar factor κ of the standard deviation (σ_r) of the distribution of the ratio metric. Mathematically, this is:

$$\Gamma_{high}(T) = \mu^{Q^r}(T) + \kappa\sigma_r \tag{2}$$

$$\Gamma_{low}(T) = \mu^{Q^r}(T) - \kappa\sigma_r \tag{3}$$

where $\Gamma_{high}(T)$ is the upper safe margin and the $\Gamma_{low}(T)$ is the lower safe margin. The conceptual details of why this is helpful can be found in [1].

2.2.3 Residuals. The idea of residuals is important for balancing missed detection and false alarms in time series anomaly detection. Residuals keep track of the difference between the safe margins and the sample $Q^r(T)$. Residual at any time window, $\nabla(T)$ equals 0, if $Q^r(T)$ is within the upper and lower safe margins and non-zero when outside the safe margins. A $Q^r(T)$ larger than the $\Gamma_{high}(T)$ (outside the upper safe margin) gives a positive $\nabla(T)$; while $Q^r(T)$ lesser than the $\Gamma_{high}(T)$ (outside lower safe margin) gives a negative $\nabla(T)$. Mathematically, this is represented as:

$$\nabla(T) : \begin{cases} = Q^r(T) - \Gamma_{high}(T), & \text{if} \quad Q^r(T) > \Gamma_{high}(T); \\ = Q^r(T) - \Gamma_{low}(T), & \text{if} \quad Q^r(T) < \Gamma_{low}(T); \\ = 0, & \text{otherwise;} \end{cases} \tag{4}$$

The $\nabla(T)$ are known as stateless residual [11], while residuals that maintain a history/time memory of residuals are known as stateful residuals. The corresponding stateful residuals are obtained from the stateless residuals by keeping the cumulative sum of the stateless residual $\nabla(T)$ over a sliding window of length F. At any time T, the stateful residual is given by:

$$RUC(T) = \sum_{j=T-FS}^{T} \nabla(j) \tag{5}$$

Thus, the vector of $RUC(T)$ values for a year will have 365 entries which can be either zero, positive or negative depending on the patterns on how stateless residuals evolve overtime during the training set without any attacks.

2.2.4 Threshold Learning from Residuals. Using the positive $RUC(T)$ values (denoted by $RUC^+(T)$) as the training data points, the best upper threshold (τ_{max}) is found. Similarly, all the negative $RUC^-(T)$ values are used as training data points for finding the best lower threshold (τ_{min}).

Setting a threshold, each from a set of discrete $RUC^+(T)$ and $RUC^-(T)$ points is similar to fitting a straight line to the set of points; hence the threshold learning problem can be viewed as a regression problem. Since the threshold is time-invariant the regression problem reduces to finding the bias term (the unknown model parameter). The model parameter search space for the upper and lower thresholds are denoted by τ^+ and τ^- respectively. Below we introduce some terminology related to regression analysis for the sake of completeness and understanding:

Error: In regression problems, the error is the difference between a model parameter candidate (candidate threshold) and a training data point. For each candidate model parameter, an error value can be calculated for any given training data point. In linear regression, positive and negative errors do not have unequal weights, and the error is usually the absolute difference.

Loss Function: The loss function is a mathematical transformation that represents *how* each error value contributes to the goodness of the candidate model parameter. For example, linear regression uses squares of the errors ($L2$ norm) as the loss function. Therefore, with respect to one candidate model parameter and one training data point, there is one loss value.

Empirical Risk Function It is an equation that maps the goodness of a candidate parameter across all training data points (observations) in the training data. Typically, it is the average/mean of the loss function values overall training data points. In linear regression, the empirical risk is called *mean squared error*. Since regression involves finding the best function that best approximates the data, the model parameter candidate that has the minimum empirical risk is the optimal answer. Hence, this type of regression is known as ordinary least squares regression.

However, [1] weighted positive and negative errors contribute unequally to the loss function. Furthermore, it used an $L1$ norm (absolute errors) instead of an $L2$ norm (squared errors) in its definition of the empirical loss function.

Algorithm 1, summarizes the basic framework for learning of thresholds as proposed in [1]. In Algorithm 1, the residual is the $\tau - RUC(T)$. Notice, that, unlike linear regression, the residuals get unequal weights and the weighted residuals are stored separately in $\mathbb{C}$ and $\mathbb{P}$.

Algorithm 1 Calculate τ_{max}

```
for RUC+(T), τ+ do
    if (RUC+(T) < τ+) then
        cost+ : |τ+ − RUC+(T)| / 2
        C ← cost+
    else
        penalty+ = |RUC+(T) − τ+|2
        P ← penalty+
    end if
end for
τ_max = arg min_{τ+} | Σ_C cost+ − Σ_P penalty+ |
```

2.2.5 Detection Criterion. The $RUC(T^{test})$ is the invariant in the test set. If it is violates the upper and lower thresholds, it indicates an attack.

$$RUC(T^{test}) : \begin{cases} \in [\tau_{min}, \tau_{max}] & \text{No Attack;} \\ \notin [\tau_{min}, \tau_{max}], & \text{Attack Inferred;} \end{cases} \quad (6)$$

Using this [1, 2] showed that small margin additive attacks show increase in the $RUC(T^{test})$ violating the upper threshold τ_{max}. For all other attack types of deductive and camouflage it was shown that the $RUC(T^{test})$ decreases violating the lower threshold τ_{min}.

3 THREAT MODEL

The main goal of poisoning attacks is to affect the learning accuracy of the detector during the training phase in such a way, that it allows false data injection to easily escape detection when launched in the test set; i.e., the adversary violates the integrity of the anomaly-based attack detector.

Naturally, this can only happen when the training attacks result in the widening of the true thresholds (upper threshold increases and lower threshold decreases). Secondly, the nature of adversarial goal dictates means the need for two threat models; one each of training and test set.

3.1 Training Phase Threat Model

Perturbation Data Type In this paper, we compare the effects of data poisoning attacks on two data types:
(1) Perturbation of Residuals: the adversary directly perturbs the residuals' set $RUC(T)$ which are direct inputs to the learning algorithm that learns the thresholds. This corresponds to the FGAV attack shown later
(2) Perturbation of Raw Data from Meters: the adversary injects false data during the training phase from a subset of smart meters, which is shown as RSL attack later.

3.1.1 Perturbation of Residuals. To change residuals directly, the adversary needs access to either (1) servers on which residual calculation happens; (2) database where the residuals are stored; and perturb them 'before' the learning of thresholds is invoked. The storage over a long time is commonly required since a large set of residuals is needed to accurately train the threshold learning model to generalize in the test set.

Perturbation Margin is the per residual change that is either added (for upper threshold) or subtracted to an input (for lower threshold), such that $0 < \epsilon < \epsilon_{max}$, $\epsilon \in \mathbb{R}^+$. The perturbation parameter is upper bounded by ϵ_{max} which is the extremum of the benign $RUC(T)$ distribution. This is motivated by making sure that the change is not out of distribution and thus imperceptible.

Fraction of Points Perturbed is the maximum no. of residual inputs that an adversary could afford to change whose upper bound is ρ_{max}. Intuitively, if ρ_{max} is low, the ϵ needs to be higher for a strong impact, and vice-versa.

Perturbation Strategy deals with which data points to select and in what order to introduce the perturbation given a ρ_{max} and ϵ. An 'adversarial example' by definition is one that does not have a random strategy of point selection, but one that uses the working weaknesses of the learning algorithm involved.

3.1.2 Perturbation of Meter Level Raw Data. It is not necessary that a utility will only face attacks from optimal adversaries. To poison the training process, an attacker can introduce false data directly from the smart meters randomly, by compromising a subset of smart meters (attack scale) and introducing some margin of false data (attack strength) per smart meter.

Training attack Strength denoted by $\delta_{avg}^{(p)}$ is the per meter average margin of false data injected during training. $\delta_{avg}^{(p)}$ is kept as a variable to check the sensitivity of the performance degradation of the anomaly detector.

Training attack Scale denoted by $\rho_{mal}^{(p)}$ is the total fraction of meters used by an adversary to launch the meter level training data poisoning. The ρ_{mal} is kept as a variable to check the sensitivity of the performance degradation of the anomaly detector.

Training Attack Type denotes how the data is falsified from its original value P_t^i which can be either additive, deductive. Intuitively, additive increases the data $P_t^i + \delta_t^{(p)}$ while deductive decreases the data such that $P_t^i - \delta_t^{(p)}$, where $\delta_t^{(p)} \in [\delta_{min}, \delta_{max}]$ and the expected value of $\delta_t^{(p)}$ samples converge to the strategic training attack strength $\delta_{avg}^{(p)}$.

While there are other attack types reported [1], in the context of poisoning to violate the integrity of the attack detector, only deductive and additive are enough. This is because it is known that small-scale additive attacks create a deviation of the residuals in the upper threshold by influencing the $RUC^+(T)$ while any deductive attack creates deviation in the lower threshold by influencing the $RUC^-(T)$. Hence, any other attack type is an overkill when the goal is achieved by a simple attack type that does not require any coordination among compromised smart meters.

3.2 Test Phase Threat Model

The test set threat model is identical to the perturbation from meter level because the whole idea is to not let the utility detect data falsification from smart meters. Hence, we refrain from elaborating this section, but only using different notations to distinguish between training and test set attacks by using two terms $\delta_{avg}^{(e)}$ (denoting attack strength used for evasion) and $\rho_{mal}^{(e)}$ (denoting attack strength used for evasion) in the test set. The above is true regardless of whether the residuals were perturbed or raw meter data was perturbed during poisoning.

The proposed contribution is divided into parts (1) Data Poisoning Attack Strategy (2) Robust Learning for Mitigation of Poisoning

4 POISONING ATTACKS STRATEGIES

In this section, we first discuss the FGAV attack followed the RSL attack.

4.1 FGAV Data Poisoning Attack

The FGAV is inspired by the Fast gradient sign method that has been shown to work well on fooling image classifiers. However, this does not apply in this setting directly, but we modified the conceptual idea of FGSM and adapted it to the strategy that applies to this grid problem.

Original Fast Gradient Sign Method: Goodfellow et. al. proposed a very efficient and generic way of producing adversarial examples to fool image classifiers that use deep neural learning frameworks to infer a wrong image class. The mathematical equation that describes the FGSM is given as:

$$x^{'} = x + \epsilon.sign(\nabla_x L(x, y)) \tag{7}$$

where $x^{'}$ is a falsified pixel intensity, and the original pixel intensity is x is modified by the perturbation ϵ. Whether the modification should be additive or deductive is dictated by the sign of the gradient (∇) of the loss function L with respect to the x given that the actual output was y.

The original FGSM method cannot be applied in our context due to the following observations:(1) it does not specify a pixel selection criterion and does this for each pixel (input). The original FGSM just optimizes which direction an input should change. However, it does not provide any procedure to select which are the best points to perturb, given an upper bound on the number of perturbation points. (2) In this case, the perturbation sign cannot be arbitrary. For missed detection increase, the perturbation sign should be positive for $RUC^{+}(T)$ inputs and negative for $RUC^{-}(T)$. *Note that, the FGAV needs to be implemented after most training data inputs are collected put the threshold learning has not yet happened.*

Fast Gradient Absolute Value (FGAV): Let the vector of input residual values (RUC) be denoted as:

$\boldsymbol{X_{RUC}} = \langle x_{ruc}(i) \rangle, \quad \forall \quad x_{ruc}(i) \rightarrow \mathbb{R}$. Each element in $\boldsymbol{X_{RUC}}$ is indexed by the set $i \in \{1, \cdots, R\}$ that uniquely identifies a residual point, such that R is the total no. of non-zero residuals recorded.

For each $x_{ruc}(i)$, we calculate the gradient with respect to the loss function L in the Algorithm 1 (which learns the threshold from residuals). This yields the following vector of gradient values $\boldsymbol{\nabla_{X_{RUC}}} = \langle \nabla_{x_{ruc}(i)} \rangle$

$$\text{where} \quad \nabla_{x_{ruc}(i)} = \frac{\partial L}{\partial x_{ruc}(i)} \rightarrow \mathbb{R}$$

Each element in the vector $\boldsymbol{\nabla_{X_{RUC}}}$ denoted by $\nabla_{x_{ruc}(i)}$ holds the signed derivative of the loss function with respect to the i-th residual point. Subsequently, we apply absolute value operation element-wise on $\boldsymbol{\nabla_{X_{RUC}}}$ and then sort its elements in decreasing order to get the following

$$\boldsymbol{\nabla^{(sort)}_{X_{RUC}}} = \langle \nabla_1^{(s)}, \cdots \nabla_j^{(s)}, \cdots \nabla_R^{(s)} \rangle \tag{8}$$

where $\boldsymbol{\nabla^{sort}_{X_{RUC}}} = \langle \nabla_j^{(s)} \rangle \quad \forall \quad \nabla_j^{(s)} \rightarrow \mathbb{R}$ and each element in $\boldsymbol{\nabla^{(sort)}_{X_{RUC}}}$ is index by the set j. We can view $\boldsymbol{X_{RUC}}$ and $\boldsymbol{\nabla^{(sort)}_{X_{RUC}}}$ as a mapping between an input set indexed by i and an output set indexed by j such that $f : i \rightarrow j$, where f is bijective. Therefore the inverse mapping $f^{-1} : j \rightarrow i$ exists.

Residual Point Selection: With the above mapping, in FGAV, the adversary selects the $i-$th residual point (co-domain in the inverse mapping) for injecting perturbation in the order that corresponds to the elements in the sorted gradient vector indexed by the set j (domain in the inverse mapping) which is the $\boldsymbol{\nabla^{(sort)}_{X_{RUC}}}$.

Perturbation per Residual: To each $i-th$ residual point picked, the perturbation margin ϵ is added to its original value $x_{ruc}(i)$, when the adversaries' objective is integrity violation of the anomaly detector; such that $x_{ruc}^{attacked}(i) = x_{ruc}(i) + \epsilon$. When the adversaries' objective is availability violation of the anomaly detector then the perturbation margin is deducted by ϵ such that $x_{ruc}^{attacked}(i) = x_{ruc}(i) - \epsilon$.

Algorithm 2 Points Selection to Violate Integrity

Input $\tau_{target}, X_{RUC}, L, \epsilon$
Output $X_{RUC}^{Attacked}$

τ_{cur} : $\arg\min_{\tau} L(X_{RUC}, \tau)$
$\nabla_{X_{RUC}} \leftarrow \frac{\partial L}{\partial X_{RUC}}$
$\nabla^{(sort)}_{X_{RUC}} \leftarrow descendingOrderOfSignedValue(\nabla_{X_{RUC}})$
while $(\tau_{cur} < \tau_{target}) \quad and \quad |X_{RUC}^{Attacked}| < \rho_{max}$ **do**
  $x_{ruc}(i) \leftarrow getMappedRUCByGradient(\nabla_{x_{ruc}}(j))$
  $x_{ruc}^{attacked}(i) \leftarrow x_{ruc}(i) + \epsilon$
  $X_{RUC} \leftarrow (X_{RUC} - x_{ruc}(i))$
  $X_{RUC} \leftarrow (X_{RUC} \bigcup x_{ruc}^{attacked}(i))$
  τ_{cur} : $\arg\min_{\tau} \frac{1}{R} \sum_R L(X_{RUC}, \tau)$
end while

Algorithm 3 FGAV Poisoning including Zero RUCs

Input $\tau_{target}, X_{RUC}, L, \epsilon$
Output $X_{RUC}^{Attacked}$

for $x_{ruc}(i)$ in X_{RUC} **do**
  if $x_{ruc}(i)$ is Zero **then**
    $x_{ruc}(i) \leftarrow x_{ruc}(i) + N(\mu, \sigma)$
  end if
end for
Call Algorithm 2

Stopping Criterion The above selection and perturbation continue until either the target is reached (in the case of targeted poisoning) or the ρ_{max} limit is reached. The whole process discussed above is summarized in Algorithm 2. The Figs. 1(a) and 1(b) show the effect of FGAV attack on the optimal value of τ_{min} for high and low value of ρ_{max} respectively.

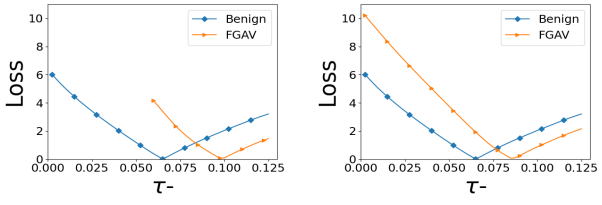
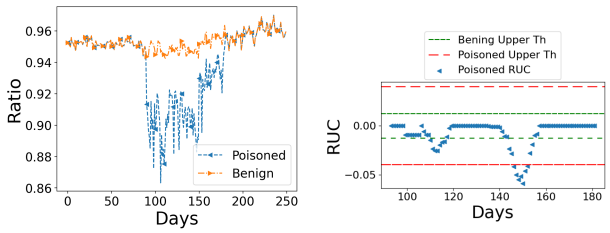


Figure 1: FGAV Poisoning Effect : (a) High ρ_{max} (b) Low ρ_{max}

4.2 Smart Meter Level Random Poisoning (RSL)

Random smart meter level poisoning gives adversaries a different advantage. It is more practical, since it can be launched anytime and the adversaries only need access to a set of CPS sensing endpoints (i.e. smart meters in our case) and no other knowledge.

Figs. 2(a) and 2(b) show the effect of meter level data poisoning on the ratio metric and eventual thresholds. Fig. 2(a) indicates that the ratio drops under a deductive poisoning attack. Hence, intuitively,



(a) (b)

Figure 2: RSL attack effect on (a) Ratio (b) Thresholds

more negative RUCs will be triggered in the $RUC^{-}(T)$ set due to the decreased ratio samples. This biases the learning of the τ_{min}. More negative RUCs trigger a lower learnt τ_{min}.

Counter-intuitively though, we also see that the learnt upper threshold τ_{max} also increases after the RSL poisoning attack, even as the ratio deviated in the direction of the lower threshold (See Fig 2(b)). This surprise is caused due to the design of the safe margins $\Gamma_{high}(T)$ and $\Gamma_{low}(T)$, which was done to lower false alarms in the original method. The ratio drop due to poisoning causes the calculated $\mu^{Q^r}(T)$ to be lower for the attack life time. Since $\Gamma_{high}(T)$ directly depends on the $\mu^{Q^r}(T)$, the $\Gamma_{high}(T)$ samples are also lowered significantly for the attack life time. However, we kept the attack life time of about 3 months of a given year. This means that in the other year, the ratio samples $Q^r(T)$ remain unperturbed and therefore a higher value compared to decreased ratio during the attack. Therefore, the $\nabla(T)$ gives a very high value for $Q^r(T) - \Gamma_{high}(T)$. Since the RSL attack is consecutive, the $RUC^{+}(T)$ values that is a cumulative sum of the $\nabla(T)$ also increases significantly for the duration of the poisoning attack. Since increased $RUC^{+}(T)$ are inputs to the learning of τ_{max}, the upper threshold is also widens in the positive direction.

From the above, we conclude that the learning of thresholds are poisoned in both directions. This is an interesting development because a single poisoning attack type (i.e. deductive) during the training can influence both the upper threshold and lower thresholds as shown in Fig. 2(b). This allows for evasion of all attack types and margins during the test set.

5 MITIGATION UNDER POISONING ATTACKS

In this section, first we give a theoretical intuition of robust learning under poisoning attacks that include M-estimation and influence function. Second, we give theoretical reasoning of our choice of M-estimator for designing robust empirical loss function by relating robustness to poisoning with influence function. Third, we propose a robust learning algorithm for thresholds using M-estimator. Finally, we discuss the learning of hyper-parameters of the robust learning algorithms. The robust learning mitigates the impact of the poisoning attack and it is in invariant of the poisoning attack.

5.1 Theoretical Intuition

We need to design a threshold learning approach that mitigates the level of bias induced in the estimate of the standard limits, as a result of the poisoning attacks. In ordinary L2 regression, the empirical loss function is $\sum_i s_i^2$, where i-th is any training data-point x_i (RUC in our case), the s_i is the error residual between the data point and a given candidate estimate $s_i = |x_i - \tau|$. The goal is to pick that candidate parameter that minimizes the empirical loss function.

To assess the vulnerability of the threshold learning to poisoning attacks, we require the rate of change in the empirical loss function (estimator) due to a small perturbation in the input data. Any design change in the learning that triggers a lesser rate of change in the empirical loss function for the same input perturbation, will allow a more robust estimate of the parameter being learned.

M-Estimator In theoretical robust statistics, the idea of a design change manifests itself by changing the definition of the loss function. Specifically, this means replacing the empirical loss function of $\sum_i s_i^2$ in classical regression with $\sum_i M(s_i)$, where $M(s_i)$ is a special function class known as M-estimator [15]. These special functions could be anything depending on the requirement, such that formally, we are solving the following

$$\tau_{opt} = \arg\min_{\hat{\tau}} \Big[\sum_{i=1}^{N} M(s_i, \hat{\tau}) \Big] \tag{9}$$

where $\hat{\tau}$ is the candidate parameter, the τ_{opt} is the optimal estimate.

Influence Function (IF) [15] is the derivative of the empirical loss function (with M-estimators) with respect to the sample inputs. This influence function gives theoretical insights into measuring the change in the estimator when sample points (training sample) are perturbed. Formally, this is defined as:

$$IF = \sum_i \frac{\partial(M(s_i))}{\partial s_i} \tag{10}$$

where s_i is directly affected by the perturbation of x_i.

In our paper, we choose two special functions to replace in the generic M-estimator form; (1) the Huber Loss and the (2) Cauchy/Lorentzian Loss and then analyze their influence functions compared to L1 and L2 used in regular regression.

5.2 Robust Loss Function as M-estimator

(1) Huber Loss [15] for regression is defined by the following theoretical expression:

$$M_{huber}(s_i) = \begin{cases} \frac{s^2}{2} & \text{if } |s| <= \beta_h; \\ \beta_h |s| - \frac{\beta_h^2}{2} & \text{otherwise.} \end{cases} \tag{11}$$

where s denotes the error between the data point and the candidate parameter, and $\beta_{(h)}$ is the scaling hyperparameter under a Huber Loss. The Huber loss is a combination of L_1 and L_2 norms as evident from the two conditional equations. For smaller errors, $s < \beta$ contribute a squared error to empirical loss, while large errors are proportional to the absolute error to the eventual empirical loss. This causes the Huber loss function to grow quadratically for errors smaller than β, and linearly for errors bigger than β.

(2) Cauchy-Lorentzian Loss [18] is a less well-known loss function whose mathematical expression is defined as:

$$M_{cauchy}(s_i) = \beta_c^2 \log\left(1 + \left(\frac{s}{\beta_c}\right)^2\right) \tag{12}$$

where s denotes the corresponding error and the β_c is the scaling hyper parameter under Cauchy Loss. Note that Eqn. 12 is a scaled L_2 norm. Since Huber loss has linear growth for large residuals compared to Cauchy, common logic would suggest using a Huber

loss over Cauchy loss for robustness. However, we found a counter-intuitive theoretical and experimental result in this case study, where Cauchy was found to be the better against poisoning attacks.

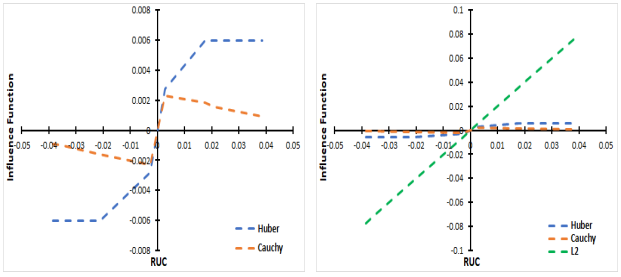
Now we explain why using Huber and Cauchy for threshold learning instead of MSE makes sense under poisoning attacks.

Influence Function Numerical Analysis The Cauchy and Huber losses are twice-differentiable, symmetric, positive definite. Therefore, by taking the derivative of Cauchy and Huber loss function, we have the following expressions:

$$IF_{Huber} = \begin{cases} s_i & \text{if } |s_i| \leq \beta_h; \\ \beta_h sgn(s_i) & \text{otherwise.} \end{cases} \tag{13}$$

$$IF_{Cauchy} = \frac{\beta_c^2(s_i)}{\beta_c^2 + {s_i}^2} \tag{14}$$

Now we use a pictorial description with an illustration in Figs. 3(a) and 3(b), to explain a key theoretical insight, which plots the influence function (in y-axis) for varying perturbation strengths on the RUC set (on the x-axis). We plotted the influence function for the L2, Huber, and Cauchy using varying perturbations to examine the rate of change of the influence function in Fig. 3(b). The green dotted line is the L2 or the MSE loss used in ordinary least squares. We can see that the rate of change of the IF for L2 is much higher compared to the other two lines IF_{huber} (blue dotted) and IF_{cauchy} (orange dotted). Since both IF_{huber} and IF_{cauchy} change at a much lower rate than L2, the difference between them is visually unclear in Fig. 3(b). Therefore, we redraw just the influence functions of Huber and Cauchy loss as M-estimator separately in Fig. 3(a). A key observation from Fig. 3(a) is at least for this numerical snapshot, the influence function of Cauchy changes at a slower rate compared to the more well-known Huber loss in robust statistics. From this theoretical insight, we expect that Cauchy will perform better in terms of robustness against poisoning attacks compared to Huber and L2 loss. This we will verify extensively in the test set.



(a) (b)

Figure 3: Influence Function versus Residual Perturbation (a) Cauchy and Huber IF (b) Cauchy, Huber and $L2$ IF

5.3 Robust Learning for Mitigation of Poisoning

Now we propose a modified quantile weighted regression and regular (unweighted) regression with robust loss functions Cauchy and Huber for learning of the thresholds to mitigate the influence of poisoning attacks, such that we have 4 new algorithms (a) Quantile Weighted Regression with Huber Loss (Algorithm 4) (b) Quantile Weighted Regression with Cauchy Loss (Algorithm 5) (c) Classical (Unweighted) Regression with Cauchy Loss (Algorithm 6); (d) Classical (Unweighted) Regression with Huber Loss (Algorithm 7).

For the algorithms, we only write the pseudocode for learning the upper threshold τ_{max}. This is because the same process applies for the learning τ_{min}. For learning τ_{min} everything in the algorithm is the same except, it runs with inputs $RUC^{-}(T)$ and parameter τ^{-}.

For the optimal learning of τ_{max}, the input training samples are $RUC^{+}(T)$ and the candidate parameter search space is τ^{+} for each algorithm. We start with explaining algorithm 4, which is based on Huber loss and uses a modified quantile weighted regression (to handle heteroskedasticity). We know that the error is the difference $\tau^{+} - RUC_i^{+}(T)$. The notion of quantile regression mandates different weights to the errors depending on whether the candidate parameter fit is greater or lesser than the input points. We felt this is necessary to reduce false alarms because the distribution of RUC samples is not Gaussian which violates the basic theoretical assumption of the ordinary least squares regression.

Algorithm 4 Quantile Weighted with Huber loss ($\tau_{max}^{(h)}$)

for $RUC^{+}(T), \tau^{+}$ **do**
  if $(RUC^{+}(T) < \tau^{+})$ **then**
    $S^{+} : |\tau^{+} - RUC^{+}(T)|\lambda_c$
    if $(S^{+} <= \beta_h)$ **then**
      $L_{s^{+}} : \left(\frac{(S^{+})^2}{2}\right)$
    else
      $L_{s^{+}} : \left(\beta_h S^{+} - \frac{(\beta_h)^2}{2}\right)$
    end if
  else
    $S^{+} : |RUC^{+}(T) - \tau^{+}|\lambda_p$
    if $(S^{+} <= \beta_h)$ **then**
      $L_{s^{+}} : \left(\frac{(S^{+})^2}{2}\right)$
    else
      $L_{s^{+}} : \left(\beta_h S^{+} - \frac{(\beta_h)^2}{2}\right)$
    end if
  end if
end for
$\tau_{max}^{(h)} = \arg\min_{\tau^{+}} \sum_{RUC^{+}(T)} |L_{s^{+}}|$

We select one candidate τ^{+} for the upper threshold, then compare whether it is greater or lesser than each point in the set $RUC^{+}(T)$. If any $RUC^{+}(T) < \tau^{+}$, we give it a λ_c weight and calculate the corresponding weighted error denoted as S^{+}, otherwise (i.e $RUC^{+}(T) \geq \tau^{+}$), we give the error a λ_p weight, such that $\lambda_p > \lambda_c$. To embedding the Huber estimator, for each weighted error S^{+}, we calculate a loss value L_s^{+} according to the Huber loss depending on whether S^{+} is greater or lesser than β_h. Note that, by the above procedure, we now have an L_s value for each training data point in the set $RUC^{+}(T)$ for a fixed τ^{+}. Then, the empirical loss value for a single candidate τ^{+} is the sum of L_s values created over each entry in the set $RUC^{+}(T)$, which is denoted in the algorithms as $\sum_{RUC^{+}(T)} L_s^{+}$. Now each candidate τ^{+} will generate a unique $\sum_{RUC^{+}(T)} L_s^{+}$. The final optimal estimate is the one τ^{+} that has the minimum $\sum_{RUC^{+}(T)} L_s^{+}$ which is the last line of the algorithm 4.

In a similar manner, we have described algorithm 5, which uses a Cauchy loss instead of Huber keeping everything else the same. Therefore, the only thing that is different is the calculation of L_s^{+} from the weighted error S^{+}. This is dictated by how Cauchy loss

Algorithm 5 Quantile Weighted with Cauchy loss ($\tau^{(c)}_{max}$)

for $RUC^{+}(T), \tau^{+}$ **do**
  if $(RUC^{+}(T) < \tau^{+})$ **then**
    $S^{+} : |\tau^{+} - RUC^{+}(T)|\lambda_c$
    $L_s+ : (\beta_c^{+})^2 \log\left[1 + \left(\frac{S^{+}}{\beta_c^{+}}\right)^2\right]$
  else
    $S^{+} : |RUC^{+}(T) - \tau^{+}|\lambda_p$
    $L_s+ : (\beta_c^{+})^2 \log\left[1 + \left(\frac{S^{+}}{\beta_c^{+}}\right)^2\right]$
  end if
end for
$\tau^{(h)}_{max} = \arg\min_{\tau^{+}} \sum_{RUC^{+}(T)} L_{s^{+}}$

mandates the transformation of the residual into a loss value. The optimal answer is denoted $\tau^{(c)}_{max}$, indicating that the used loss function was Cauchy for learning the threshold.

For the case where there is no asymmetric weights given to the errors, Algorithms 6 and 7 have been re-written for Cauchy and Huber respectively using a similar logic. Note that, in each case, we are only learning the bias parameter without the slope, since we want a fixed time-independent threshold for flagging an anomaly.

Algorithm 6 Regular Regression with Cauchy loss ($\tau^{(c)}_{max}$)

for $RUC^{+}(T), \tau^{+}$ **do**
  $S^{+} : |\tau^{+} - RUC^{+}(T)|$
  $L_s+ : (\beta_c^{+})^2 \log\left[1 + \left(\frac{S^{+}}{\beta_c^{+}}\right)^2\right]$
end for
$\tau^{(c)}_{max} = \arg\min_{\tau^{+}} \sum_{RUC^{+}(T)} \log L_{s^{+}}$

Algorithm 7 Regular Regression with Huber loss ($\tau^{(c)}_{max}$)

for $RUC^{+}(T), \tau^{+}$ **do**
  if $(S^{+} <= \beta_h)$ **then**
    $L_{s^{+}} : (\frac{(S^{+})^2}{2})$
  else
    $L_{s^{+}} : (\beta_h S^{+} - \frac{(\beta_h)^2}{2})$
  end if
end for
$\tau^{(h)}_{max} = \arg\min_{\tau^{+}} \sum_{RUC^{+}(T)} |L_{s^{+}}|$

5.4 Learning Hyperparameters

Now we discuss about selection of the scaling hyperparameter of the loss functions, β_c, β_h, the hyperparameter of the weighted quantile regressor namely λ_c and λ_p over a cross validation set. The search space of β_c and β_h is bounded between 0 and the extremum in the set of S^{+} or S^{-} and the $1 > \lambda_p > \lambda_c > 0$.

We partitioned the 2016 dataset's first three months and used several values of β scaling hyperparameter whose feasible range is dictated by the feasible range of errors (i.e. $|RUC - \tau|$) as discussed in the previous subsection. We tried several combinations of β in each partition and picked the best one in each partition by using the *following* objective in Eqn. 15 and then took the average.

$$\underset{\beta_c^{+},\beta_c^{-},\lambda_c,\lambda_p}{\arg\min} \quad (w_{md} MD + w_{fa} FA) \tag{15}$$

For all results, we kept $w_{fa} = 0.7$ and $w_{md} = 0.3$. The weight assignment is reasoned by the fact that reducing the false alarm is at least more than half as important. We minimize the joint false alarm (FA) and missed detection rate (MD) in the cross validation sets. We use a random search, to find the approximate best choices of the hyperparameter. We consider this as one potential weakness in the paper which we will seek to improve in our future work with an efficient hyper-parameter search technique.

5.5 Security Evaluation Metrics

Here we explain the two main security evaluation metrics.

Impact of Undetected Attack (I): The effectiveness of integrity violation is quantified through the impact of the undetected attack on the utility after using our attack and defense [12]. To calculate the impact of undetected attack per day, we use lost revenue in terms of the unit price of electricity denoted by *RR* which is calculated as:

$$RR = (\delta_{avg} * M * \eta * E)/1000 \tag{16}$$

where the $\eta = 24$ is the number of reports per day, $E = \$0.12$ is the average per unit (KW-Hour) cost of electricity in USA, δ_{avg} is the margin of false data in test set, M is the number of compromised smart meters in the test set. The impact of undetected attack is:

$$Impact = RR * T_{detect} \tag{17}$$

T_{detect} is the time difference between attack start and end of test set in days. The lesser the impact, the better the performance.

Expected Time between Consecutive False Alarms: The expected time between two false alarms $E_{T_{FA}}$ is a recent NIST [12] recommended metric for time series anomaly detection:

$$E_{T_{FA}} = \frac{\sum_{\eta_{FA}} T_{BFA}}{\eta_{FA} - 1} \tag{18}$$

where η_{FA} is the number of false alarms and the T_{BFA} is the time gap between any pair of consecutive false alarms. If there is one false alarm, then this equation does not exist, and instead of $E_{T_{FA}}$ is 364. The logic is that another false alarm is likely to be seen in a year. The lesser the $E_{T_{fa}}$, the worse is the performance.

6 EXPERIMENTAL EVALUATION

For the experimental evaluation, a real AMI (Advanced Metering Infrastructure) data set from the Pecan Street project [19] containing power consumption records for 200 smart meters for three years (2014,2015,2016) is used. These smart meters records are divided into three parts for training, cross-validation, and testing. The training data contains all the records from the years 2014 and 2015 as was the case in previous work [1] to keep a fair performance. The remaining records from the year of 2016 data are divided into two parts for cross-validation and testing. From the year 2016, all the records within the first three months are used for cross-validation the remaining records from the later 9 months are used as the test set. The hyperparameter β_c and β_h are learned through cross validation which are equal to 0.0002 and 0.0001 respectively.

Rest of this section is divided into two subsections: 1) RSL 2) FGAV. Under RSL, for each loss function and regression type, we measure the performance in terms of an Impact of Undetected Attack as a function of the following attack variables: 1) Poisoning Attack Strength δ^{p}_{avg}; 2) Evasion Attack Strength δ^{te}_{avg} 3) Attack Scale ρ_{mal} and offer conclusions.

Similarly, under FGAV, the performance for different loss function and regression type in terms of Impact of Undetected attack is measured considering the following variables: 1) Poisoning Margin ϵ 2) Evasion Attack Strength δ^{te}_{avg} and 3) Attack Scale α_{max}, ρ_{mal}. We also compare the expected time between false alarms for different methods. Additionally, we report the expected time between two false alarms when there are no poisoning attacks at all.

Table 2: Acronyms for Design Choices

Acronym	Regression	Loss Function
QC	Quantile	Cauchy
QH	Quantile	Huber
NQC	Regular	Cauchy
NQH	Regular	Huber
L1	**Quantile**	MAE
L2	**Quantile**	MSE

6.1 Experimental Validation under RSL Poisoning

RSL Poisoning Implementation Details: For RSL poisoning attack, we varied poisoning margin δ^{p}_{avg} between 50 – 300 and varied ρ^{p}_{mal} between 20%-70% and the RSL attack duration was between 1st Apr-30th June, 2014.

RSL Evasion Implementation Details: To investigate performance in the test set, we crafted attacks with varying percentage of meters $\rho^{(te)}_{mal}$, and varying data perturbation margin δ^{te}_{avg}. The combinations were selected such that none of the attacks violate the learned standard limit to get the impact of an undetected attack. In the test set, we varied $\rho^{(te)}_{mal}$ between 20%-70%, and δ^{te}_{avg} between 10-400, which cover a wide range of attack intensities.

To remove bias in performance, we launched evasion attacks on three distinct time frames in the year 2016, each of length 90 days separately. The reported IMPACT (USD) in every result is the *average* impact of undetected attack per 90 day frame, across three attack time frames.

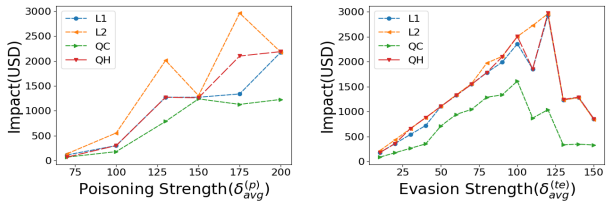
Performance evaluation of Robustness: For comparative performance evaluation, we compare all loss function choices with 2 groupings (a) within quantile weighted regression and (b) within unweighted regression. We pick the best performing loss function from each group and then compare their performance to finally conclude which loss function type and what regression type is most robust for a given attack.

The performance is measured in terms of the mean impact of undetected attack per 90 days (I) (when there is an evasion attack in the test set) and the expected time between false alarms ($E_{T_{fa}}$) (when there is no attack throughout the test set).

The impact of an undetected attack is affected by two things; (i) the poisoning attack strength (ii) the evasion attack strength required to remain undetected. Therefore, we use the above two attack parameters to measure the sensitivity of the impact of undetected attacks across various loss functions and regression choices.

6.1.1 Impact Within Quantile Regression Group: Here, we discuss the loss function choices but with the asymmetric weighing of positive versus negative residuals. Fig. 4(a), indicates that across all possible poisoning attack strengths $\delta^{(p)}_{avg}$, the QC is either equal to or lower in attack impact across all loss function choices. The evasion attack strength used is $\delta^{(te)}_{avg} = 100$ and scale is $\rho_{mal} = 0.60$.

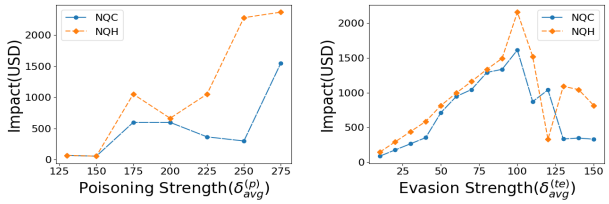
Next we vary the evasion strength to assess performance. In Fig. 4(b), we observe that across various evasion attack strengths, the QC is lower in impact than all other loss function choices. The poisoning attack strength is $\delta^{(p)}_{avg} = 200$ and scale is $\rho_{mal} = 0.50$ for Fig. 4(b).



(a) (b)

Figure 4: Impacts under Quantile Regression:(a) Varying Poisoning Attack Strength (b) Varying Evasion Attack Strength

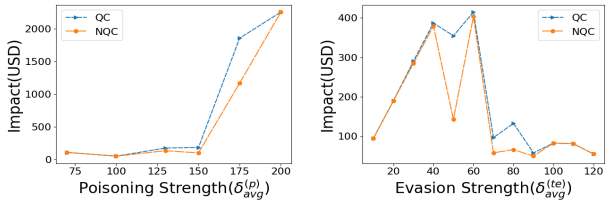
6.1.2 Within Ordinary Regression Group: Here, we discuss the loss function choices but with no weighing difference between positive versus negative residuals. Fig. 5(a), indicates that across all possible poisoning attack strengths $\delta^{(p)}_{avg}$, the NQC (unweighted Cauchy Loss) is either equal to or lower in impact than NQH . The evasion attack strength is $\delta^{(te)}_{avg} = 150$ and scale is $\rho_{mal} = 0.40$. In contrast, Fig. 5(b), indicates that across all possible evasion strengths $\delta^{(te)}_{avg}$, the NQC either is lower in impact than NQH. The poisoning attack strength is $\delta^{(p)}_{avg} = 200$ and scale is $\rho_{mal} = 0.50$ for this figure.



(a) (b)

Figure 5: Impacts under Ordinary Regression: (a) Varying Poisoning Attack Strength (b) Varying Evasion Attack Strength

6.1.3 Final Performance and Recommendation: We compare the winning loss function under quantile regression and regular regression and compare it here. We find that the regular version of Cauchy is better in terms of minimizing the impact of an undetected attack. However, this comes at a cost of increased false alarms.



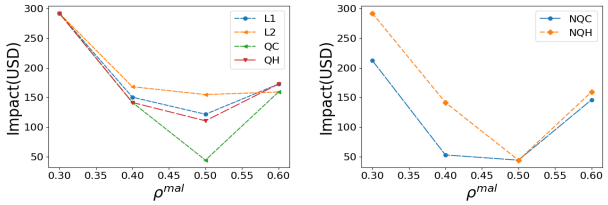
(a) (b)

Figure 6: Quantile versus Regular Regression with Cauchy Loss: (a) Poisoning Strength (b) Evasion Strength

Fig. 6(a) shows that the impact of NQC is either equal or lower than the impact of QC for varying poisoning strength δ^{p}_{avg} for a

representative evasion attack with $\delta^{te}_{avg} = 150$ and $\rho_{mal} = 0.30$. Similarly, Fig. 6(b), shows either equal or lower impact as performance for NQC than the impact of QC for varying evasion attack for a representative poisoning attack with $\delta^{p}_{avg} = 130$ and $\rho_{mal} = 0.50$. The results clearly indicates that NQC loss function outperforms all other loss functions in terms impact of undetected.

6.1.4 Effect of Varying Attack Scale: Our intention, here is to double-check whether and how the previous conclusions are affected by changing attack scale. We want to check whether any and how the above conclusions are changed if you change the ρ_{mal}.



(a) (b)

Figure 7: Impact versus Attack Scale: (a) QR (b) NQR

Fig. 7(a), shows that the QC is either equal to or lower in impact than all other loss functions.Similarly,Fig. 7(b), shows that the NQC is either equal to or lower in impact than NQH. For both Fig 7(a),Fig. 7(b), the poisoning attack strength is $\delta^{(p)}_{avg} = 100W$ and the evasion strength is $\delta^{(te)}_{avg} = 120W$.

6.1.5 Expected Time Between False Alarm (RSL). To measure the expected time between false alarms, we used the annual base rate of false alarms with no attacks throughout the year of 2016 in the test set. This is because it is worst in total absence of any attack. Table 3 presents the expected time between false alarm for different loss functions in the presence of RSL poisoning attack.

Empirical Loss	$E_{T_{fa}}$
L1	334.29
L2	364.65
QC	115.21
QH	313.84
NQC	103.53
NQH	253.01

Table 3: Expected Time between False Alarm: RSL Poisoning

Conclusion We conclude that Cauchy is the best loss function from the perspective of impact mitigation when poisoning happens. The non-quantile version of Cauchy is better in terms of minimizing the impact of undetected attacks but is worse in terms of false alarm performance (103 days) compared to the Quantile version of Cauchy (115 days). The conclusions match with the theoretical intuition.

6.2 Experimental Validation under FGAV Poisoning

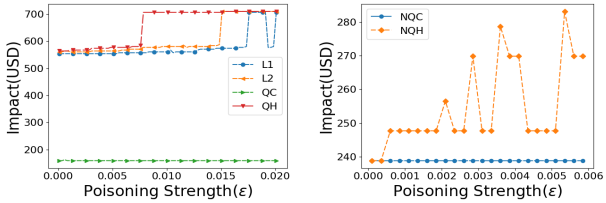
FGAV Poisoning Implementation Details: In the FGAV poisoning attack scheme, we applied perturbation ϵ, (bounded by the maximum RUC value in the training set) and the number of perturbed points bounded by α_{max}. We introduce perturbation according to Algorithm 2 for Small Scale (SS) and Medium Scale (MS) Perturbation and Algorithm 3 for Large Scale (LS) perturbation, where scale depends on the value of α_{max}. This is a time agnostic attack scheme that exploits the transformed low-dimensional input from a time series to replicate poisoning similar in principal with image attacks.

FGAV Evasion Implementation Details: To investigate performance in the test set, we crafted attacks following the similar process used for RSL evasion attack implementation described in 6.1.

Performance evaluation of Robustness: The comparative robustness of different loss functions depends on the impact of undetected I and the expected time between false alarm $E_{T_{fa}}$ just like the RSL scheme. However, here the impact of undetected attack is affected by two things; (i) the poisoning attack strength (ϵ) (ii) the evasion attack strength (δ^{te}_{avg}). We show the robustness for three different poisoning attack scales which depends on the number of perturbed residual inputs α_{max}.

6.2.1 Small number perturbations (SS):. In this section, the performances in terms of the impact of undetected are measured and compared when the number of inputs that can be perturbed by the adversary is bounded by a small number.

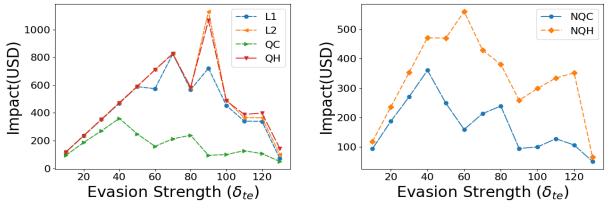
Impact vs Poisoning Strength (ϵ): Fig 8(a) is showing that the impact of QC is either equal to or lower than the impact of all other loss functions. Similarly, Fig. 8(b), is showing that the impact of NQC is either equal to or lower in impact than NQH. For Fig 8(a), the parameters have values such as $\alpha_{max} = 4$, $\delta^{te}_{avg} = 100$, and $\rho_{mal} = 0.30$. Also, In Fig. 8(b), the parameters have values such as $\alpha_{max} = 4$, $\delta^{te}_{avg} = 80$, and $\rho_{mal} = 0.30$



(a) (b)

Figure 8: *(SS):* Impact I vs ϵ: (a) Quantile Regression (b) Unweighted Regression

Impact versus Evasion δ^{e}_{avg}: This section presents how the performances of each of the loss functions are affected for varying evasion attack for fix poisoning strength.



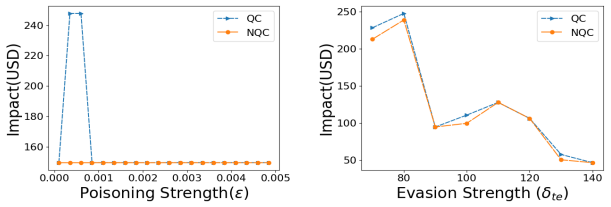
(a) (b)

Figure 9: *SS:* Impact I vs Evasion Strength (a) All Quantile Regression (b) Ordinary Regression

Fig 9(a) is showing that the impact of QC is either equal to or lower than the impact of all other loss functions. Similarly, Fig. 9(b), is showing that the impact of NQC is either equal to or lower in impact than NQH. For both Fig 9(a),Fig. 9(b), the parameters have values such as $\alpha_{max} = 4$,$\epsilon = 0.0156$, and $\rho_{mal} = 0.60$

Quantile Vs Non-Quantile: Here,the winner of the quantile loss

functions and the winner of the non-quantile loss function are compared to get the ultimate winner in terms of impact of undetected.



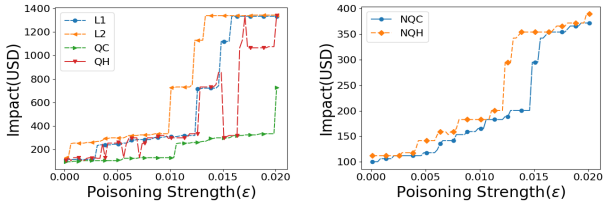
(a) (b)

Figure 10: *SS:* Quantile versus Regular Regression with Cauchy Loss (a) Poisoning Attack(b) Evasion attack

Fig 10(a) shows that impact of NQC is either equal or lower than the impact of QC for varying ϵ with evasion attack $\delta^{te}_{avg} = 70$, $\alpha_{max} = 4$, and $\rho_{mal} = 0.40$. Similarly, Fig 10(b) indicates that the impact of NQC is either equal or lower than the impact of QC for varying evasion attack with $\alpha_{max} = 4$,$\epsilon = 0.00060$ and $\rho_{mal} = 0.40$.

6.2.2 Medium number perturbations (MS):. We measure performance for different loss functions in terms of impact when the adversary has access to more input points that he can exploit to introduce perturbation than Case 1.

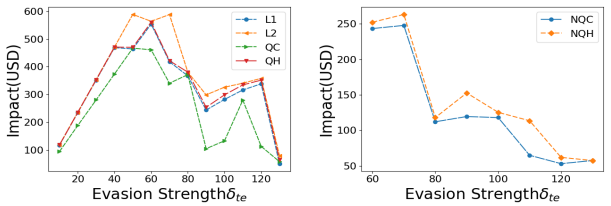
Impact vs Poisoning Strength (ϵ): Fig 11(a) is showing that the impact of QC is either equal to or lower than the impact of all other loss functions. Similarly, Fig. 11(b), is showing that the impact of NQC is either equal to or lower in impact than NQH. For Fig 11(a), the parameters have values such as $\alpha_{max} = 10$, $\delta^{te}_{avg} = 90$, and $\rho_{mal} = 0.30$. Also, In Fig. 11(b), the parameters have values such as $\alpha_{max} = 16$, $\delta^{te}_{avg} = 80$, and $\rho_{mal} = 0.40$



(a) (b)

Figure 11: *(MS:)* Impact I for varying poisoning ϵ (a) Quantile Regression (b) Ordinary Regression

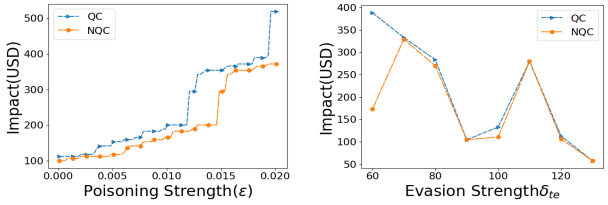
Impact versus Evasion δ^{e}_{avg}: Fig 12(a) is showing that the impact of QC is either equal to or lower than the impact of all other loss functions. Similarly, Fig. 12(b), is showing that the impact of NQC is either equal to or lower in impact than NQH. For both Fig 12(a),Fig. 12(b), the parameters have values such as $\alpha_{max} = 12$,$\epsilon = 0.00485$, and $\rho_{mal} = 0.30$



(a) (b)

Figure 12: *MS:* Impact I for Evasion Strength (a) All Quantile Regression (b) Ordinary Regression

Quantile versus Non-Quantile Here, the winner of the quantile loss functions and the winner of the non-quantile loss functions are compared to get the ultimate winner in terms of the impact of undetected.



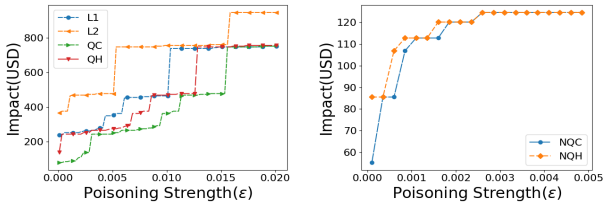
(a) (b)

Figure 13: *MS:* Impact of Quantile versus Regular Regression with Cauchy Loss (a) Poisoning Attack (b) Evasion attack

Fig 13(a) shows that impact of NQC is either equal or lower than the impact of QC for varying ϵ with evasion attack $\delta^{te}_{avg} = 80$, $\alpha_{max} = 16$, and $\rho_{mal} = 0.40$. Similarly, Fig 13(b) indicates that the impact of NQC is either equal or lower than the impact of QC for varying evasion attack with $\alpha_{max} = 16$,$\epsilon = 0.0036$ and $\rho_{mal} = 0.30$.

6.2.3 Large number perturbations (LS):. In this section the performance, when an adversary can perturb a large number of input points utilizes Algorithm 3 of the FGAV attack.

Impact vs Poisoning Strength (ϵ): Fig 14(a) is showing that the impact of QC is either equal to or lower than the impact of all other loss functions. Similarly, Fig. 14(b), is showing that the impact of NQC is either equal to or lower in impact than NQH. For Fig 14(a), the parameters have values such as $\alpha_{max} = 32$, $\delta^{te}_{avg} = 60$, and $\rho_{mal} = 0.40$. Also, In Fig. 14(b), the parameters have values such as $\alpha_{max} = 28$, $\delta^{te}_{avg} = 20$, and $\rho_{mal} = 0.20$

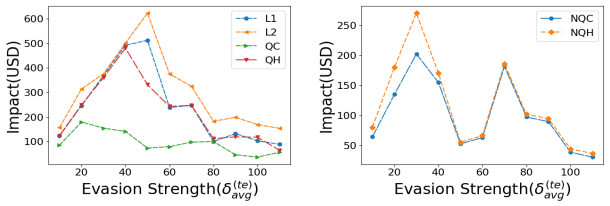


(a) (b)

Figure 14: *(LS):* Impact I vs poisoning strength ϵ (a) All Quantile Regression (b) Ordinary Regression

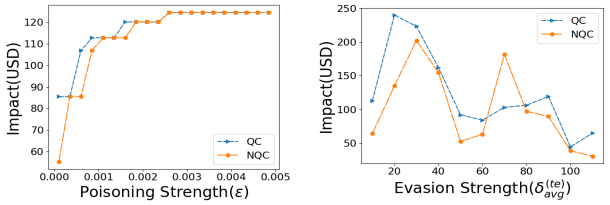
Impact versus Evasion δ^{e}_{avg}: This section will show the compared performances under varying evasion attack. Fig 15(a) is showing that the impact of QC is either equal to or lower than the impact of all other loss functions. Similarly, Fig. 15(b), is showing that the impact of NQC is either equal to or lower in impact than NQH. For both Fig 15(a),Fig. 15(b), the parameters have values such as $\alpha_{max} = 32$, $\epsilon = 0.00035$, and $\rho_{mal} = 0.40$

Quantile versus Regular Here, the winner of the quantile loss functions and the winner of the non-quantile loss function are compared to get the ultimate winner in terms of the impact.



(a) (b)

Figure 15: *LS*: Impact vs Evasion Strength for loss functions (a) All Quantile Regression (b) Ordinary Regression



(a) (b)

Figure 16: *LS:* Quantile vs Regular Regression with Cauchy Loss across (a) Poisoning Strength (b) Evasion Strength

Fig 16(a) shows that impact of NQC is either equal or lower than the impact of QC for varying ϵ with evasion attack $\delta^{te}_{avg} = 20$, $\alpha_{max} = 40$, and $\rho_{mal} = 0.20$. Similarly, Fig 16(b) indicates that the impact of NQC is either equal or lower than the impact of QC for varying evasion attack with $\alpha_{max} = 32$,$\epsilon = 0.00060$ and $\rho_{mal} = 0.40$. Considering the similarity in each cases,it is evident that NQC is recommended in terms of the impact of undetected performances.

6.2.4 Expected Time Between False Alarm for FGAV:. Table 4 shows the effective time between false alarm for different methods in the presence of FGAV poisoning attack.

Empirical Loss	Avg.$E_{T_{fa}}$	SS $E_{T_{fa}}$	MS $E_{T_{fa}}$	LS $E_{T_{fa}}$
L1	337.47	364.20	364.20	340.10
L2	364.35	364.44	364.42	364.30
QC	247.66	304.02	304.02	258.43
QH	309.66	364.02	304.02	303.70
NQC	244.36	303.93	257.95	257.95
NQH	257.09	304.00	287.67	258.43

Table 4: False Alarm Performance: FGAV Poisoning

6.3 Expected Time between False Alarm without Poisoning

Table 5 shows the expected time between false alarm results when there is no poisoning; and also there is no evasion attack during testing. It shows that the use of robust loss function comes with a price of increased false alarm. The base rate probability of occurrence of a poisoning attack is not high. Therefore, while deploying a model with robust loss function this trade-off must be considered and the weights during the cross validation may be tuned according to the needs of a provider.

Conclusions on FGAV: We find similar patterns as RSL. The cauchy non-quantile is best in terms of impact robustness. The use of non-quantile causes slightly worse false alarm performance for all FGAV cases, which matches with our theoretical intuition.

Empirical Loss	$E_{T_{fa}}$
L1	364.0
L2	364.0
QC	45.5
QH	364.0
NQC	45.5
NQH	45.5

Table 5: False Alarm Performance: No Poisoning Attacks

7 CONCLUSIONS

In this paper, we showed two ways of poisoning anomaly-based attack detectors for smart metering infrastructure and compared their effects. The use of L1 norm instead of the L2 norm (i.e., OLS regression) for threshold learning, gives some protective effect against FGAV. This is because the L1 norm induces an effect called gradient shattering which does not allow the calculation of accurate gradients. We verified this in the experimental section where we showed that the impact of poisoning attacks for L2 norm is greater than L1 norm. We proposed threshold learning for anomaly detection with robust loss function under quantile and unweighted regression. We analyzed security performance with impact of undetected attacks and expected time between false alarms. For impact robustness, we found that the Cauchy is the better loss function choice than more known Huber loss. Accounting for base rate expected time between false alarms, we found that the quantile weighted regression is a better choice than regular regression.

Acknowledgements: The research was supported by USA's NSF grants, OAC-2017289 and SATC-2030611.

REFERENCES

[1] S. Bhattacharjee, S.K. Das, "Detection and Forensics against Stealthy Data Falsification in Smart Metering Infrastructure", *IEEE Trans. of Dependable and Secure Computing*, Vol. 18, Jan. 2021.
[2] S. Bhattacharjee, P. Madhavarapu, S. Silvestri, S. K. Das, "Attack Context Embedded Data Driven Trust Diagnostics in Smart Metering Infrastructure" *ACM Transactions on Privacy and Security*, Apr. 2021.
[3] D. Urbina, J. Giraldo, A. Cardenas, N. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell and H. Sandberg, "Limiting the Impact of Stealthy Attacks on Industrial Control Systems", *ACM CCS)*, pp. 1092-1105, 2016.
[4] P. Roy, S. Bhattacharjee, S.K. Das, "Real Time Stream Mining based Attack Detection in Distribution Level PMUs for Smart Grid", *IEEE Globecomm* 2020.
[5] S. Bhattacharjee, A. Thakur, S.K. Das, "Towards Fast and Semi-Supervised Identification of Smart Meters launching Data Falsification", *ACM Asia CCS*, 2018.
[6] M. Wilbur, A. Dubey, B. Leao, S. Bhattacharjee, "A Decentralized Approach For Real Time Anomaly Detection In Transportation Networks , *IEEE SMARTCOMP)*, 2019
[7] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples", *ICLR*, 2015.
[8] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *ICLR* 2017.
[9] F. Tramer, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *ICLR* 2018.
[10] P. Roy, S. Bhattacharjee, S.K. Das, "Real Time Stream Mining based Attack Detection in Distribution Level PMUs", *IEEE GLOBECOM*, Dec. 2020.
[11] D. Urbina, J. Giraldo, A. A. Cardenas, N. Tippenhauer, J. Valente, M. Faisal, R. Candell, H. Sandberg, "Limiting the Impact of Stealthy Attacks on Industrial Control Systems" *ACM CCS*, 2016.
[12] D. Urbina, J. Giraldo, A. A. Cardenas, N. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, H. Sandberg, "Survey and New Directions for Physics-Based Attack Detection in Cyber-Physical Systems", *ACM Computing Surveys*, Vol. 51(10), 2018.
[13] G. Jiraldo, A. Cardenas, " Adversarial Classification under Differential Privacy", *NDSS*, 2020.
[14] N. Dalvi P. Domingos, M. Mausam, S. Sanghai, D. Verma "Adversarial Classification" *ACM SIGKDD*, 2004.
[15] P. Huber, E. Ronchetti, "Robust Statistics", *Wiley Probability and Statistics*, ISBN 978-0470129906, Feb. 2009.
[16] M. Jagliesk, A. Oprea, C. Liu, C. Rotaru, B. Li, "Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning", *IEEE S & P Oakland*, 2018.
[17] A. Ghafouri, Y. Vorobeychik, X. Koutsoukos, "Adversarial Regression for Detecting Attacks in Cyber-Physical Systems" *IJCAI*, 2018.
[18] X. Li, Q. Lu, Y. Dong and D. Tao, "Robust Subspace Clustering by Cauchy Loss Function," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 30(7), pp. 2067-2078, 2019.
[19] [Online] Pecan Street Project Dataset https://www.pecanstreet.org/
[20] [Online] https://www.globalsign.com/en/blog/cyber-autopsy-series-ukranian-power-grid-attack-makes-history