MALICIOUS DATA DETECTION IN STATE ESTIMATION LEVERAGING SYSTEM LOSSES & ESTIMATION OF PERTURBED PARAMETERS

BY

WILLIAM NIEMIRA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Master's Committee:

Professor Peter Sauer
Research Assistant Professor Rakesh Bobba

ABSTRACT

It is critical that state estimators used in the power grid output accurate results even in the presence of erroneous measurement data. Traditional bad data detection is designed to perform well against isolated random errors. Interacting bad measurements, such as malicious data injection attacks, may be difficult to detect. In this work, we analyze the sensitivities of specific power system quantities to attacks. We compare real and reactive flow and injection measurements as potential indicators of attack. The use of parameter estimation as a means of detecting attack is also investigated. For this the state vector is augmented with known system parameters, allowing both to be estimated simultaneously. Perturbing the system topology is shown to enhance detectability through parameter estimation.

To my family, advisors, instructors, and classmates

ACKNOWLEDGMENTS

This research was supported by the Department of Energy under Award Number DE-OE0000097 (TCIPG).

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Operators of the power grid rely on large networks of sensors, known as SCADA (Supervisory Control and Data Acquisition) systems, to monitor conditions on the grid. To incorporate the many redundant measurements available into a single coherent picture, the grid state is estimated from the output of these sensors periodically. The state estimates affect both operational and economic functions, so it is critical that state estimates accurately reflect the grid state.

State estimators are classified as DC state estimators, utilizing a linear system model, or AC state estimators, using a nonlinear model. Under the DC model, typical measurements consist of real power flows and injections and states consist of bus angles [1].

Traditional bad data detectors are designed primarily to deal with random sensor noise or errors. Potential sources for noisy or erroneous measurements could be non-simultaneity of measurement sampling, incorrect installation of or damage to sensor, or communication errors. These schemes are designed to detect isolated, random bad data. Interacting bad data is more difficult to detect. A subclass of interacting bad data is malicious data injection attacks.

Malicious data injection attacks against state estimators were first written about in [2]. These attacks consist of coordinated modifications to measurements, such that modifications are coherent with the linearized model and unmodified measurements. This conceals the presence of attacks from DC state estimators completely. The effect of linear model attacks on nonlinear estimators was shown to be diminished in [3]; however [4] showed the potential for DC attacks to succeed on real EMS software using a nonlinear model.

This paper examines the sensitivity of real and reactive power measurement residuals in a nonlinear state estimator to false data injection attacks based on a linearized model. The effects of malicious data injection attacks

are shown to have varying degrees of effect on different measurement types and that residuals of some measurement types are a much better indicator of attacks. This paper also presents a detector based on parameter estimation combined with topology perturbation of estimated parameters.

CHAPTER 2

BACKGROUND

2.1 State Estimation

State estimation is the use of measurements of a system to estimate unknown system quantities. This allows a complete description of a system to be developed from various redundant measurements. Accuracy can be improved, because although the individual measurements are subject to error, the impact of random errors in measurements is reduced by measurement redundancy. In addition to potentially improving accuracy, state estimation allows estimation of quantities that are difficult to measure directly, such as phase angle.

The simplest form of state estimation for power systems uses a linearized model to estimate bus voltage angles based on real power flows. This linearized, or DC, approach neglects real series impedances, all shunt impedances, real losses, and all reactive power. A flat voltage profile is also assumed. Linearization decreases computation time, and may yield acceptable results. The linear relationship between measurements z and states x under the DC model is shown in Equation (2.1). H is a matrix containing the linearized model information. H resembles the admittance matrix under these model assumptions. The disadvantage of linearization is a loss of information. The linearized state estimator will not give any information about the nonlinear quantities neglected, including any reactive flows and injections, or bus voltage magnitudes. A state estimator using a nonlinear system model, or AC state estimator, is necessary to obtain information about nonlinear quantities. For the AC case, H is a vector function relating measurements z to states x as in Equation (2.2).

$$z = Hx \tag{2.1}$$

$$z = H(x) \tag{2.2}$$

2.2 Parameter Estimation

System parameters can be estimated at the same time as system states by augmenting the state equations. In the case of state augmentation, H becomes a vector function relating measurements z to states x and parameters p as in Equation (2.3). For a linearized model, this augmentation changes Equation (2.1) to Equation (2.4). The partition H_x in Equation (2.4) is the original H matrix from Equation (2.1). In both cases, this means that the number of equations to be solved simultaneously increases by the number of additional parameters. This approach to parameter estimation through state augmentation is one of the approaches from [1], and was the approach used for this research.

$$z = H(x, p) \tag{2.3}$$

$$z = \begin{pmatrix} H_x & H_p \\ 0 & I \end{pmatrix} \begin{pmatrix} x \\ p \end{pmatrix} \tag{2.4}$$

2.3 Bad Data Detection

State estimation uses models to produce estimates; essentially fitting data to a model. Traditional bad data detection generally assumes that measurement errors will be large and isolated, resulting in a poor fit to the model. Detection of isolated bad data has been addressed in many works. As examples, a method using normalized residuals is addressed in [5] and another based on the coherency between measurements with large residuals and the other measurements in [6].

Detection of multiple bad data using various methods has also been investigated. Examples include [7], where a linear program was used to detect multiple bad data sources that did not fit a Gaussian noise model, and [8], where hypothesis testing was used for multiple and interacting bad data de-

tection.

2.4 Malicious Data Attacks and Detection

Data injected deliberately by an adversary would not be expected to follow the same patterns as random bad data. A method of injecting bad data into linearized state estimation (DC model) without changing measurement residuals was demonstrated in [2] and [9]. These attacks are not detectable by traditional bad data detection schemes accompanying linear state estimators and are dubbed stealth attacks. To construct these attacks, the attacker needs some knowledge of the system topology.

It was shown in [10] that full knowledge of system topology is not necessary, and also that the attack based on the linearized model can be effective against a non-linear estimator. Although [3] shows decreased effectiveness of linear attacks when used against non-linear estimators, [4] shows that real EMS software is indeed vulnerable to attack.

A data injection attack detection scheme, based on hardening of sets of measurements, is shown in [11]. Measurements to be hardened are selected such that observability is guaranteed for the operator, which is shown to be necessary and sufficient to prevent stealth attacks. A Bayesian detector for malicious data injection attacks is formulated in [12]. The efficient use of encryption to thwart attacks is investigated in [13], where an algorithm to maximize the utility of encrypted measurement placement is developed.

A financial incentive for attacks on state estimators is described in [14], where it is shown that the use of malicious data injection by an adversary can manipulate energy markets such that an attacker can guarantee trading profits.

2.5 Topology Perturbation

A topology perturbation based detector for detection of malicious data is developed in [15]. That approach uses known topology perturbations (specifically changes to impedance) applied to a power system as probes. The expected system response is predicted and compared against the actual mea-

surements. The topology perturbations are hidden from an attacker, so the attacker is unable to correct the attack to remain stealthy. If measurement values do not change as computed beforehand, an attack is assumed. Although effective, this method has the disadvantage that it has to assume no changes in load and generation between probes or account for such changes making it difficult. The approach here is to estimate the parameter changes with measurement data rather than to predict measurement changes. Furthermore instead of probing the system, the estimate of the parameters can be used for detection by leveraging regularly occurring perturbations.

CHAPTER 3

SYSTEM LOSS AND PARAMETER BASED DETECTION

3.1 Attacker Model

The attacker is assumed to have access to baseline topology information necessary to formulate a DC attack offline, i.e., at least one of the columns of the sensitivity matrix used for linear state estimation. This information is assumed to be static, and will not reflect changes such as line outages, tap changes, or other topology or parameter changes. The attacker is also assumed to have the ability to change a subset of measurements needed for an attack. The change could occur by corrupting the measurement device at the substation, interfering with communication between the substation and control center, or by installation of malware at the control center. The attacker is assumed not to have the ability to observe conditions across the entire system.

3.2 Nonlinear Sensitivity Analysis

3.2.1 Overview

The sum of squared residuals of a state estimator's output quantifies how well the measurement data fits the model. This is useful for bad data detection, as bad data would not be expected to fit the model well. Unlike bad data, malicious data is designed to fit a model, albeit a simplified or partial model, decreasing the impact on measurement residues. Using knowledge of the DC model, we can predict what measurements would tend to be impacted most by an attack. This work focuses on real and reactive power measurements. Two classes of measurements, flows and injections, are considered here.

The DC model neglects the effects of losses, uneven voltage profiles, and reactive power on the power system. When an AC state estimator is attacked using attacks based on a DC model, the residues increase due to these simplifications. The magnitude of effect on real and reactive flows and injections varies by measurement type.

For real flows, the DC model tends to underestimate their magnitude due to neglecting real power losses. The impact of this simplification increases with the square of current on a line. Reactive flows also have losses that increase with the square of line current, however the high X/R ratio of power lines means that reactive power transmission is inherently lossier than real power transmission. When combined with the DC models neglect of reactive power, the expected effect is for malicious data injection attacks to affect reactive flow residues more than real flow residues.

The error from neglecting losses becomes more apparent when generator power output is monitored. Losses have to be supplied by a generator, so errors on flows due to neglecting losses will be accumulated at generators.

To verify these intuitions, and to examine the detectability of malicious data injection attacks, the effect of attacks of various magnitudes on a composite weighted residual, and on residues for real flows, reactive flows, real generator injections, and reactive generator injections was investigated.

3.2.2 Establishing Baseline

Baseline residual values had to be established for comparison purposes. Measurement residues are expected to vary under normal circumstances due to noise. To establish the distribution of residuals expected due to this expected noise, Monte Carlo trials were used. To do this, noise vector n was added to measurement vector z, resulting in z^* as seen below in Equation (3.1).

$$z^* = z + n \tag{3.1}$$

State estimation was conducted for many such random noise vectors, and sums of squared residues of real and reactive flows and generator injections were recorded separately and together as a weighted composite.

Using the distributions generated in this fashion, a threshold residue value can be established. For values higher than this, malicious data is assumed.

The cutoff is selected based on the acceptable proportion of false alarms. For example, if 1% false alarms are acceptable, the cutoff chosen would be a residue value higher than the residue for 99% of trials.

3.2.3 Comparison of Detectability

To determine the detectability of a particular attack vector at a particular magnitude, the Monte Carlo procedure used in the previous section was repeated for a system under attack. Noise vector n and attack vector a were added to measurement vector z, resulting in z^{**} as seen below in Equation (3.2).

$$z^{**} = z + n + a \tag{3.2}$$

The detectability of an attack by a residual can be quantified by the proportion of measurements above the baseline cutoff established earlier. For example, if 80% of the distribution of residuals for the attacked system lies above the cutoff, that attack would be considered 80% detectable. This allows comparison of the effectiveness of residuals of specific measurement types for detecting bad data, as well as comparing the detectability of different attacks.

The total detectability is the non-overlapping proportion of attacks detected by any of the metrics. This may be larger than detectability based on any single residual type.

3.3 Parameter Estimation

3.3.1 Overview

The use of parameter estimation to detect maliciously injected data was also investigated. The state estimator was modified for this purpose. State vector x augmented with parameters p to form $\tilde{x}$ as in Equation (3.3). The equation used to relate system measurements and states becomes a function of parameters as well, as shown in Equation (3.4).

$$\tilde{x} = \begin{pmatrix} x \\ p \end{pmatrix} \tag{3.3}$$

$$\tilde{z} = h(\tilde{x}) \tag{3.4}$$

And where $\tilde{z}$ is the measurement vector z augmented with parameter estimates $\bar{p}$ as in Equation (3.5).

$$\tilde{z} = \begin{pmatrix} z \\ \bar{p} \end{pmatrix} \tag{3.5}$$

The sensitivity matrix H, which is the function $h(.)$ as evaluated at each iteration is also modified as shown in Equation (3.6).

$$\tilde{H} = \begin{pmatrix} H_x & H_p \\ 0 & I \end{pmatrix} \tag{3.6}$$

The parameters estimated could be any system parameter, for instance, transformer tap position or capacitor bank setting. In this work, line series reactance was considered as a parameter to estimate. Distributed flexible AC transmission system (D-FACTS) devices were assumed to have been installed on the system. This allows line impedance to be controlled by the control center, and might be done for congestion management and loss minimization.

Installation of D-FACTS devices turns line reactances into variable parameters. If parameter values change, i.e. the D-FACTS setting is altered, attacks based on pre-change values will perform poorly. If an attack is constructed for a system using the wrong parameter values, the interactions between attacked measurements tends to drive parameter estimates toward the values used to construct the attack vector. This increases parameter estimate residues.

The residues on parameter estimates are more meaningful for the purpose of malicious data detection than measurement residuals because the true value of the parameter can be known with more precision. The parameter value, unlike measurements, should not be affected by grid state. This makes comparison of estimated parameters with their known values useful for malicious data detection.

Detection of malicious data injection attacks is possible whenever the pa-

rameters are altered for control purposes and the alteration can be concealed from the attacker at least for a while.

3.3.2 Establishing Baseline

As with measurement residuals in Section 3.2.2, baseline parameter residual values had to be established for comparison purposes. To do this, noise vector n was added to measurement vector $\tilde{z}$, resulting in $\tilde{z}^*$ as seen below in Equation (3.7). Noise was limited to measurement values (not parameters).

$$\tilde{z}^* = \tilde{z} + n \tag{3.7}$$

3.3.3 Determining Detectability

The procedure of Section 3.2.3 was repeated to generate distributions of parameter residuals. To do this, measurement vector $\tilde{z}$ was augmented with noise vector n, and attack vector a, resulting in $\tilde{z}^{**}$ as seen below in Equation (3.8).

$$\tilde{z}^{**} = \tilde{z} + n + a \tag{3.8}$$

The effects of perturbations on detectability can be determined by comparing distributions generated for the same attack vectors under different system perturbations to each other and to the base case.

CHAPTER 4

EVALUATION

4.1 Nonlinear Sensitivity Analysis

4.1.1 Setup

Analysis was conducted on the IEEE 14-bus test case. State estimation was executed using MATPOWER, a MATLAB power system simulation package. Each noise vector n used for generating distributions in Monte Carlo trials had entries consisting of a normally distributed random variable with zero mean and standard deviation of 1% of the measurement value corresponding to the entry in n. Distributions of measurement residuals for determining baseline values and attack detectability were generated by executing state estimation using 500 different noise vectors. The cutoff used to establish the baseline as described in the previous section was the 99th percentile (i.e. 1% false alarms).

4.1.2 Establishing Baseline

Data consisting of sums of squared residues of real and reactive flows and injections were recorded separately and as a weighted composite. Histograms of the result were generated from this data. An example for real power flows, is shown below in Figure 4.1. From these histogram bin counts, a cumulative density function plot was created by normalizing the histogram to have an area of 1 and computing the cumulative sum of bin counts. The 99% cutoff was established by finding the sum of squared residue value of the CDF at the 99th percentile. An example for reactive power flows is shown below in Figure 4.2.

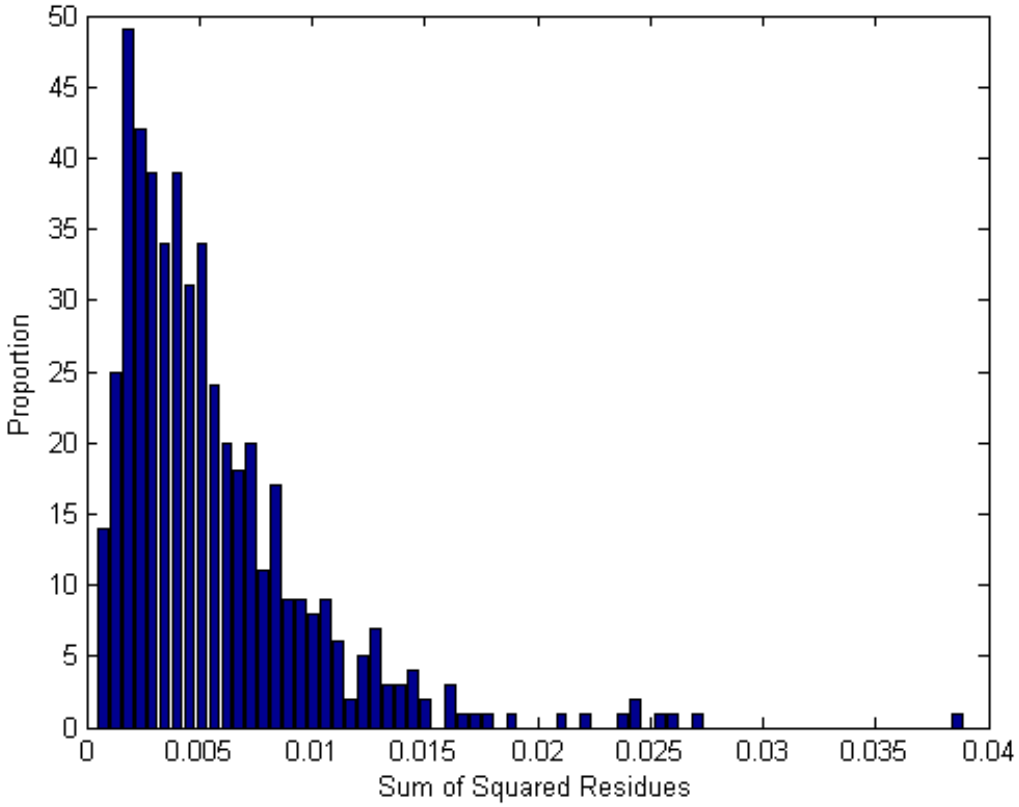

Figure 4.1: Histogram of sum of squared residues of reactive power flows for 500 random noise vectors.

4.1.3 Determining Detectability

Distributions of sums of squared residues were constructed for the system under various attack regimes. In theory any linear combination of columns of the DC H-matrix can serve as a stealth attack vector. For our experiments we used individual columns of the DC H-matrix (14 in all) albeit scaled as attack vectors.

Attack vectors were scaled such that the largest entry in each attack vector corresponded to a 10MW injection (0.1 p.u. in 100 MVA base). Subsequent trials increased the scale of attack vectors in 10MW steps up to 100MW, so that each of the 14 columns of the DC H-matrix was used at each of 10 different power levels (10 to 100MW in 10MW increments). For every distribution generated, the percentage of measurements falling above the previously determined residual threshold was calculated.

It was predicted that generator injections, due to having to supply losses, would be impacted most by malicious data injection attacks. Reactive flows

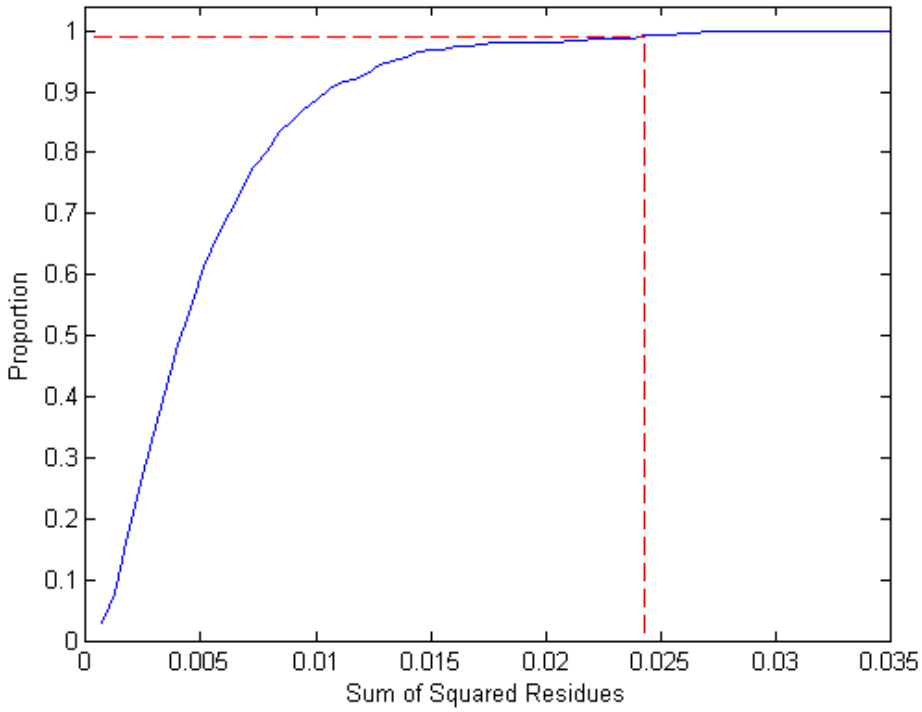

Figure 4.2: CDF of sum of squared residues of reactive power flows for 500 random noise vectors. Horizontal line indicates 99th percentile. Vertical line indicates value for sum of squared residues corresponding to 99th percentile.

were also expected to be impacted more than real flows. To compare the relative sensitivities to attack, the residual with the highest proportion of detectable attacks was recorded for each of the 14 attack columns at each magnitude (140 in total). Table 4.1 contains counts of the number of attacks for which a residual type is best, based on having the highest detectability for that attack.

The grouped bar graph of Figure 4.3 shows the proportion of attacks detected by each of the residual types at the 10MW attack level. The total detectability is also shown, which indicates the proportion of attacks that caused any one of the residuals to exceed its cutoff value. This is higher than any residual type taken singly. As expected both Table 4.1 and Figure 4.3 show that there is significant difference in the residuals based on measurement type and that residuals of real and reactive power injections are better at indicating the presence of attack data.

Figure 4.4 shows the total detectability sampled at attack levels of 30MW,

Table 4.1: Detection by Residual Type

Residual Type	**Attacks Detected Best**
Weighted Composite	2
Real Power Flows	8
Real Power Injections	60
Reactive Power Flows	17
Reactive Power Injections	53

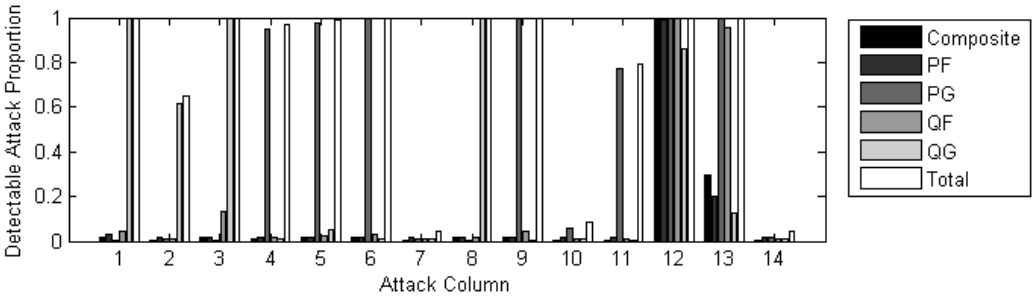

Figure 4.3: Grouped bars indicating the proportion of each attack detected at the 10MW attack level. Bars in each group from left to right are residuals of: the weighted composite residual, real power flow, real power generation, reactive power flow, reactive power generation, and total detectability.

50MW, 80MW, and 100MW. At 30MW, most of the attacks exceed one of their residue thresholds and are detectable. Attacks based on columns 7, 10, and 14 in the DC H-matrix have the poorest detectability. And while attacks based on column 7 begin to have good detectability from 50MW upward, attacks based on columns 10 and 14 were still undetectable even at attack magnitude of 80MW. As a reference the total injections in the system are about 275MW. Attack based on column 14 reamind only partially detectable even at attack magnitude of 100MW. Buses 10 and 14 were sparsely connected, each only connecting to two other buses, and were not connected to any buses with generation. These attacks required the least amount of measurement modifications, and their distance from generation minimized the impact on generator residuals.

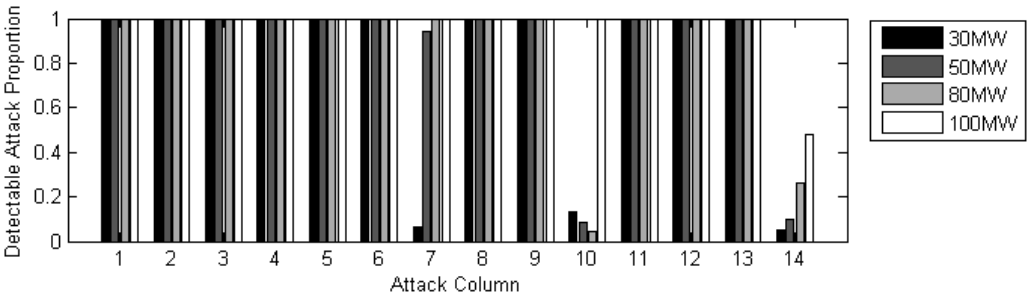

Figure 4.4: Grouped bars indicating the total proportion of each attack column type detected at 30MW, 50MW, 80MW, and 100MW, attack levels.

4.2 Parameter Estimation

4.2.1 Setup

As was done previously, analysis was conducted on the IEEE 14-bus test case. State estimation was executed using a modified version of MATPOWER. The modified program estimates parameters simultaneously with states using state augmentation as described in Section 3.3.1, and was used in this case to estimate line series reactances. Each noise vector n used for generating distributions in Monte Carlo trials had entries consisting of a normally distributed random variable with zero mean and standard deviation of 1% of the measurement value corresponding to the entry in n. Distributions of measurement residuals for determining baseline values and attack detectability were generated by executing state estimation using 500 different noise vectors. The residual threshold used to establish the baseline as described in the previous section was the 99th percentile (i.e. 1% false alarms). D-FACTS devices were added to branches 4, 8, and 12, allowing the series reactance to be modified by 5%. As before, attacks consist of the columns of the DC H-matrix used in DC state estimation. Attack magnitude was not varied. Instead, attacks were all normalized at the 100MW level.

4.2.2 Establishing Baseline

To establish expected distributions of parameter estimate residues due to measurement noise, the state estimation was performed with the addition of random noise vectors on measurements. The threshold used to establish the

baseline as described in the previous section was 99%.

Histograms of the sum of squared errors of parameter estimates were generated from this data. From these histogram bin counts, a cumulative density function plot was created by normalizing the histogram to have an area of 1 and computing the cumulative sum of bin counts. The 99% cutoff was established by finding the sum of squared residue value of the CDF at the 99th percentile.

4.2.3 Determining Detectability

Distributions of residuals were created for parameter residues when the system was unperturbed and under attack, and for attacks against a perturbed system. These distributions were compared to the baseline residuals to determine the effects of attacks on parameter residuals, and to see whether topology perturbation in combination with parameter estimation enhances the detectability of malicious data injection attacks.

Figure 4.5 shows a comparison of parameter residue distributions. The unattacked system has the lowest residues, and is shown at the far left of the plot. For this attack, the parameter estimates for an unperturbed system would be a good indicator, as we see that most of the area under the CDF for the attacked, unperturbed system (dot-dash) falls above the cutoff established for normal data. The effect of perturbation in this case is to shift the CDF even higher, indicating improved detection of attacks. This effect is also present at lower attack levels, albeit to a lesser extent, as seen for an attack at the 50MW level in Figure 4.6.

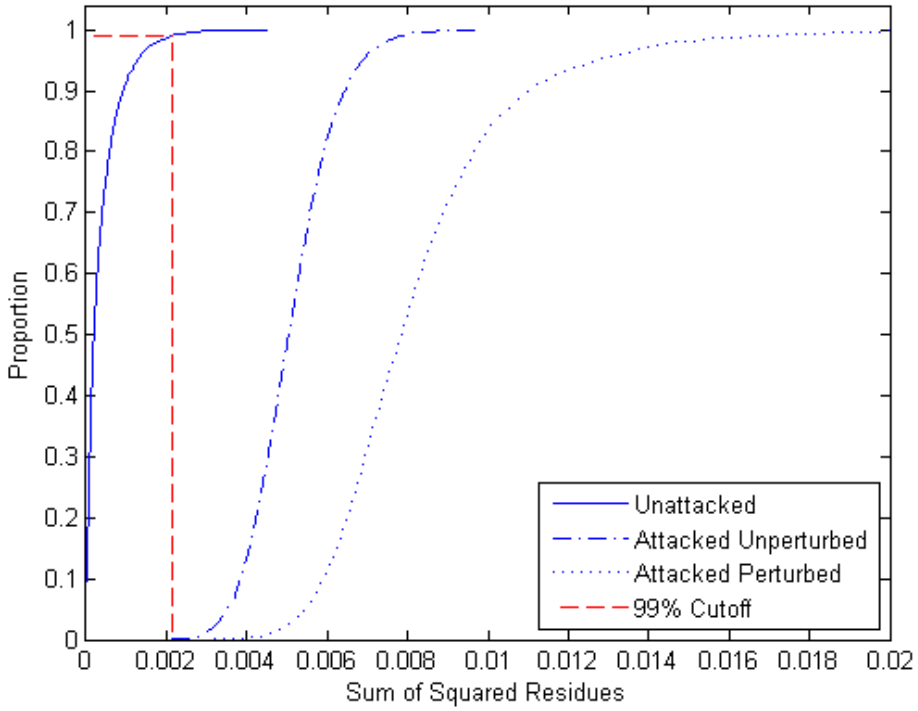

Figure 4.5: CDF of sum of squared residues of parameter estimates for unattacked system (solid), attacked system without topology perturbation (dot-dash), and with perturbation (dotted). Horizontal line (dashed) indicates 99th percentile under normal conditions. Vertical line (dashed) indicates value for sum of squared residues corresponding to 99th percentile. Attacks were at 100MW level.

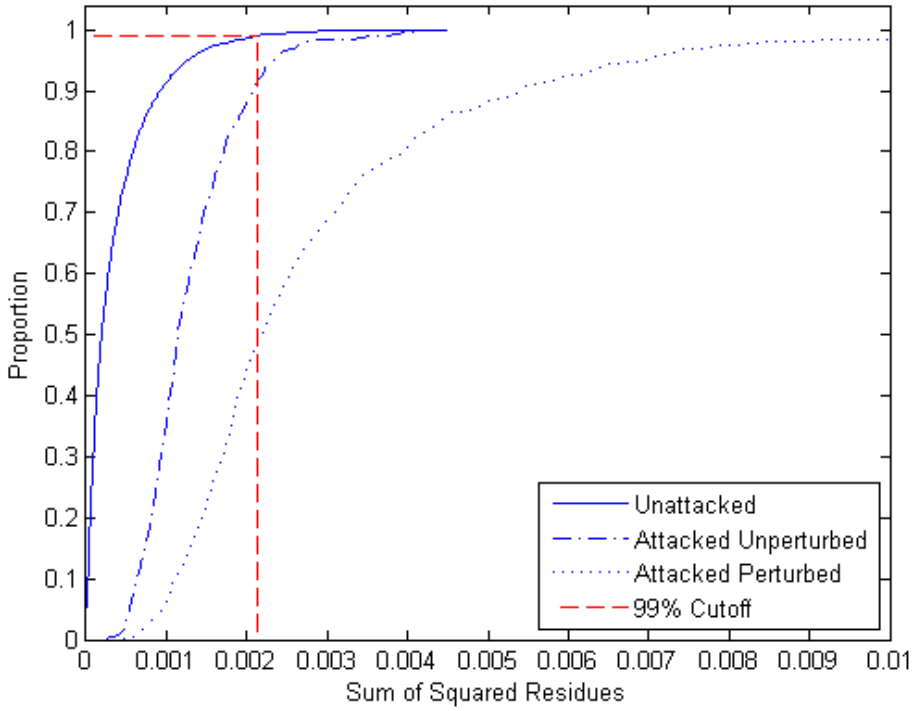

Figure 4.6: CDF of sum of squared residues of parameter estimates for unattacked system (solid), attacked system without topology perturbation (dot-dash), and with perturbation (dotted). Horizontal line (dashed) indicates 99th percentile under normal conditions. Vertical line (dashed) indicates value for sum of squared residues corresponding to 99th percentile. Attacks were at 50MW level.

CHAPTER 5

CONCLUSION

The simplifying assumptions made by an attacker using a linear model to construct attacks against a non-linear state estimator do not affect all measurement types equally. The DC model was shown to introduce very little residue on real power flow measurements in comparison to generators. Generators, which must supply losses, accumulate the effects of attacks. We established in this work that malicious data injection attacks should be expected to impact some classes of measurements greater than others and that such differences can be leveraged to detect maliciously injected data.

Parameter residues were also found to increase with data injection attacks. Perturbation of the system in conjunction with parameter estimation was shown to cause further increase. This increase enhances detectability of malicious data injection attacks.

We also found that some attacks are hard to detect even at high energy levels. We intend to study these types of attacks further in the future for different bus systems and identify ways to detect them. Future work could include comparisons of residual distributions to ordinary bad data to distinguish between bad and malicious data. Another area for future research is locating which measurements specifically are being attacked if malicious data injection occurs.

CHAPTER 6

REFERENCES

[1] A. Monticelli, *State Estimation in Electric Power Systems: a Generalized Approach*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999.

[2] Y. Liu, M. K. Reiter, and P. Ning, "False data injection attacks against state estimation in electric power grids," in *Proc. of the 16th ACM Conference on Computer and Communications Security*, 2009.

[3] L. Jia, R. Thomas, and L. Tong, "On the nonlinearity effects on malicious data attack on power system," in *Power and Energy Society General Meeting, 2012 IEEE*, 2012, pp. 1–8.

[4] A. Teixeira, G. Dán, H. Sandberg, and K. H. Johansson, "Cyber security study of a scada energy management system: Stealthy deception attacks on the state estimator," in *18th IFAC World Congress, Milan, Italy*, 2011.

[5] E. Handschin, F. Schweppe, J. Kohlas, and A. Fiechter, "Bad data analysis for power system state estimation," *Power Apparatus and Systems, IEEE Transactions on*, vol. 94, no. 2, pp. 329–337, 1975.

[6] A. Monticelli and A. Garcia, "Reliable bad data processing for real-time state estimation," *Power Apparatus and Systems, IEEE Transactions on*, vol. PAS-102, no. 5, pp. 1126–1139, 1983.

[7] W. Peterson and A. Girgis, "Multiple bad data detection in power system state estimation using linear programming," in *System Theory, 1988., Proceedings of the Twentieth Southeastern Symposium on*, 1988, pp. 405–409.

[8] L. Mili, T. Van Cutsem, and M. Ribbens-Pavella, "Hypothesis testing identification: A new method for bad data analysis in power system state estimation," *Power Engineering Review, IEEE*, vol. PER-4, no. 11, pp. 31–32, 1984.

[9] Y. Liu, M. K. Reiter, and P. Ning, "False data injection attacks against state estimation in electric power grids," in *ACM Trans. in Information and Systems Security (TISSEC)*, 2011.

[10] A. Teixeira, S. Amin, H. Sandberg, K. Johansson, and S. Sastry, "Cyber security analysis of state estimators in electric power systems," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, 2010, pp. 5991–5998.

[11] R. Bobba, K. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T. Overbye, "Detecting false data injection attacks on dc state estimation," 2010.

[12] O. Kosut, L. Jia, R. Thomas, and L. Tong, "Limiting false data attacks on power system state estimation," in *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, 2010, pp. 1–6.

[13] G. Dán and H. Sandberg, "Stealth attacks and protection schemes for state estimators in power systems," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 214–219.

[14] L. Xie, Y. Mo, and B. Sinopoli, "False data injection attacks in electricity markets," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 226–231.

[15] K. Morrow, E. Heine, K. Rogers, R. Bobba, and T. Overbye, "Topology perturbation for detecting malicious data injection," in *System Science (HICSS), 2012 45th Hawaii International Conference on*, 2012, pp. 2104–2113.

[16] R. Zimmerman and C. Murillo-Sánchez, "Matpower user's manual." [Online]. Available: http://http://www.pserc.cornell.edu/matpower/

[17] R. Bo, "Brief introduction to state estimation program." [Online]. Available: http://http://www.pserc.cornell.edu/matpower/

[18] I. The MathWorks, "New rng function in r2011a." [Online]. Available: http://www.mathworks.com/videos/new-rng-function-in-r2011a-69034.html

APPENDIX A

USING MATPOWER STATE ESTIMATION FOR RESEARCH

A.1 Overview

MATPOWER is a powerful and flexible tool for power system research. It is open source, and available free online with documentation [16]. The state estimation portion of MATPOWER, located in `matpower4.1/extras/se` was used both in stock and modified form for the research in this paper. Some brief documentation specific to the state estimator is available in [17]. This appendix will provide some instruction for using MATPOWER for state estimation and state estimation augmented with parameter estimation to contnue research in this area.

A.2 Generating New Cases

To conduct research on state estimation, it is necessary to generate cases to study. This section describes how to do this efficiently.

A.2.1 Systems

MATPOWER comes bundled with files for systems ranging from a few buses to thousands of buses. There may still be need to import systems for research purposes, particularly if specific features are required for the research (for instance, TCUL transformers, DFACTS devices, capacitor banks, shunt reactors, etc.). MATPOWER uses a proprietary system data format, however it is similar in structure to the IEEE common data format (`.cdf` extension) and MATPOWER comes with a function for converting from the IEEE common data format to MATPOWER format.

The function `cdf2matp.m` can output a MATPOWER system case file given a file in the IEEE common data format. This is convenient, because it allows importation of sytems from any other program capable of saving to the IEEE common data format, or the use of freely available test cases in the IEEE format. Unfortunately, the IEEE common data format standard is looser than `cdf2matp.m` expects, which can cause parsing errors. During the course of research for this paper, several systems were constructed in and imported from PowerWorld. Initially, some of the cases would not properly import due to header data being longer than allowed for in `cdf2matp.m`. This caused MATLAB give an error related to "dimension" of a string or matrix; parsed text was longer than allocated string length. Reducing the number of characters in the header line (manually, with a text editor) fixed this problem.

A.2.2 Measures

In addition to system information, measurement sets are needed to conduct state estimation. Measurements could be obtained by running a simulation in PowerWorld or other power system simulator and manually entering the data into the case file. This is cumbersome, and prone to errors. Instead, it is better to use MATPOWER's built in functions to obtain measurements.

For this work, measurement sets were generated by solving the power flow problem, and then saved as MATLAB vectors. The function `runpf.m` runs a power flow, and can output the data needed for state estimation. This requires returning results as individual output arguments like this:

```
[baseMVA, bus, gen, branch, success, et] = runpf(casedata);
```

The desired measures can be found in `bus` (voltage measurements), `gen` (generator injections), `branch` (line flows). Indicies can be defined to make it easier to extract the desired measurement types from here by running the following code.

```
[PQ, PV, REF, NONE, BUS_I, BUS_TYPE, PD, QD, GS, BS, BUS_AREA,...
    VM, VA, BASE_KV, ZONE, VMAX, VMIN, LAM_P, LAM_Q, MU_VMAX,...
    MU_VMIN] = idx_bus;
```

```
[F_BUS, T_BUS, BR_R, BR_X, BR_B, RATE_A, RATE_B, RATE_C, TAP,...
    SHIFT, BR_STATUS, PF, QF, PT, QT, MU_SF, MU_ST] = idx_brch;

[GEN_BUS, PG, QG, QMAX, QMIN, VG, MBASE, GEN_STATUS, PMAX,...
    PMIN, MU_PMAX, MU_PMIN, MU_QMAX, MU_QMIN] = idx_gen;
```

Measurement vectors are then extracted from `bus`, `gen`, and `branch` using the named indicies. For example, the vector of power flow measurements measured at the "from" end of the line would be saved to the variable `PFvalues` like this:

```
PFvalues = branch(:, PF);
```

These measures can then be inserted into a state estimation case file. See `test_se.m` which comes with the MATPOWER state estimator for an example of how the file is formatted. The state estimation case file consists of five sections. The first section specifies which measurements are to be used (measurement indicies), this is followed by the values of measurements, measurement variances, a data integrity check, and a call to `run_se`.

In addition to measurement indicies, values, and variances, the user must supply the number of buses (as the variable `nbus`) and the name of the file containing the system data (as the variable `casename`). The state estimation case file is also a good place to augment measures with noise from a pseudorandom number generator and/or with an attack vector. The "seed" value for the pseudorandom number generator can be used to control for specific random noise regimes when comparing different attacks, allowing two different attacks to be tested against the same noise distributions.

A.2.3 Attacks

The function `DCH.m` was created to generate attack vectors. An entire DC H-matrix is generated in the file, and then the specified column is returned. The entries in the DC H-matrix are generated from sums of line reactances, but attention must be paid to signs depending on the direction of the line.

A.3 Parameter Estimation

For this research, modified versions of MATPOWER were created that added network parameter estimation to state estimation. Initially modified to estimate transformer tap positions, code was eventually generated for line reactance and resistance as well. These versions of MATPOWER have the directory names `matpower4.1_TAP`, `matpower4.1_Xs`, and `matpower4.1_Rs` respectively. The differences between these MATPOWER flavors is merely the sensitivity functions called in `doSE.m`. Because the tap position estimator was written first, the other versions may still have variables containing the word "tap", even though a different parameter is actually being estimated. It may be desirable to modify MATPOWER in the future to allow selection of parameter to be estimated by passing an input parameter. (e.g. Calling `run_se` with the parameter "t" would execute state estimation with transformer tap position estimation. The parameter "x" would cause line reactances to be estimated, and so on.) This would reduce the amount of code to maintain.

Although it was not done for this work, it is possible to estimate multiple types of parameters at once. However, it was not trivial to get a single parameter type to work properly, and adding more will complicate things.

A.3.1 Algorithm

The algorithm used to perform parameter estimation is based on the state augmentation, described in [1] on page 289 and also here in Sections 2.2 and 3.3.1.

Table A.1: Functions in MATPOWER State Estimation

Function	**Purpose**
`run_se.m`	"Main" function, containing calls to the other functions
`checkDataIntegrity.m`	Checks input data for completeness
`isobservable.m`	Checks for observability
`doSE.m`	Computes state estimates using an iterative process
`outputsesoln.m`	Prints the resuts of state estimation

A.3.2 Modifications to MATPOWER

MATPOWER state estimation is conducted using the functions discussed in Table A.1. Of these, the functions modified to augment states with parameters is in Table A.2. The functions `checkDataIntegrity.m` and `isobservable.m` were not critical and were left unmodified. If problems are encountered because of this (and the problem is not a genuine one) calls to these functions can be commented out. The file `run_se.m` did not need to be modified, as it only contains calls to other functions. To create the H-matrix, functions capable of returning sensitivities (partial derivatives) of measurements with respect to parameters are required. These functions are similar to the functions `dSbr_dV.m` and `dSbus_dV.m`, and are listed in Table A.3.

Table A.2: Functions modified for parameter estimation

Function	**Modification**
`doSE.m`	Outside the main loop many variables were augmented to include parameters. Inside the main loop matrix H was modified to include sensitivites of measures to parameters, and modifications were made to compute new parameter estimates and update values.
`outputsesoln.m`	Modified to print parameter estimates.
state estimation case files	When creating case files, variables for measurement indicies, values, and variances must include fields for parameters.

Table A.3: Functions created for parameter estimation

Function	**Purpose**
`dSbr_dTAP/Xs/Rs.m`	Returns sensitivities of real and reactive flows (PF, PT, QF, QT) w.r.t. TAP/Xs/Rs
`dSG_dTAP/Xs/Rs.m`	Returns sensitivities of real and reactive injections (PG, QG) w.r.t. TAP/Xs/Rs
`dV_dTAP/Xs/Rs.m`	Returns sensitivities of voltage (Va, Vm) w.r.t. TAP/Xs/Rs

A.3.3 Repetitive Trials

The state estimation case file containing measurement values was modified to add input parameters for simulations requiring repetitive trials. Inputs added were `col` (column number from the DC H-matrix corresponding to an attack), `a` (attack magnitude scaling factor), `n` (noise magnitude scaling factor), and `seed` (seed value for pseudorandom number generator). The function `rng(seed)` was introduced in MATLAB R2011a (described in [18]). Use of this seeding function makes R2011a or newer required to use this method.

For running trials, a script can then call the modified function in a loop allowing the attack space to be explored systematically under many different attack and noise realizations. For this research, nested loops were used to test each of the attack vectors possible at varying magnitudes for many different noise realizations. This requires a loop for the attack column, a loop for the attack scaling, and a loop for the noise selection.

Function calls to generate the attacks and noise vectors were made inside the case file based on parameters passed by the trial script. The actual generation of noise and attack vectors was performed each time the state estimation case file was called. This means that many of the same noise vectors were generated multiple times under different attacks. For performance, it may have been adventageous to pass the actual attack and noise vectors, rather than parameters specifying them. This would have allowed noise and attack vectors to be precomputed and stored, saving computation time per trial. This was not done for the sake of expediting code development.

In MATLAB, printing output to the screen slows performance considerably. To speed up repetitive trials, the output results portion of `run_se.m` were suppressed via commenting. Similarly, integrity and observability checking of data were suppressed by comment. While it may be desirable to run checks on a measurement case file one time, there is no need to check the integrity of the same case file at each iteration; if the check is passed once, it will be passed every time.