

University of Arkansas, Fayetteville

ScholarWorks@UARK

---

Computer Science and Computer Engineering  
Undergraduate Honors Theses

Computer Science and Computer Engineering

---

5-2021

## Data Forgery Detection in Automatic Generation Control: Exploration of Automated Parameter Generation and Low-Rate Attacks

Yatish R. Dubasi

Follow this and additional works at: <https://scholarworks.uark.edu/csceuht>



Part of the [Artificial Intelligence and Robotics Commons](#), [Controls and Control Theory Commons](#), [Data Storage Systems Commons](#), [Information Security Commons](#), [Other Computer Sciences Commons](#), [Power and Energy Commons](#), [Signal Processing Commons](#), and the [Systems and Communications Commons](#)

---

### Citation

Dubasi, Y. R. (2021). Data Forgery Detection in Automatic Generation Control: Exploration of Automated Parameter Generation and Low-Rate Attacks. *Computer Science and Computer Engineering Undergraduate Honors Theses* Retrieved from <https://scholarworks.uark.edu/csceuht/88>

This Thesis is brought to you for free and open access by the Computer Science and Computer Engineering at ScholarWorks@UARK. It has been accepted for inclusion in Computer Science and Computer Engineering Undergraduate Honors Theses by an authorized administrator of ScholarWorks@UARK. For more information, please contact [ccmiddle@uark.edu](mailto:ccmiddle@uark.edu).

**Data Forgery Detection in Automatic Generation Control: Exploration  
of Automated Parameter Generation and Low-Rate Attacks**

An Undergraduate Honors College Thesis

in the

Department of Computer Science and Computer Engineering

College of Engineering

University of Arkansas

Fayetteville, AR

by

Yatish Reddy Dubasi

Thesis title: Data Forgery Detection in Automatic Generation Control: Exploration of Automated Parameter Generation and Low-Rate Attacks

Thesis author: Yatish Dubasi

The thesis is approved by:

Thesis Advisor:


Signature:  Date: 4/12/2021

Printed: Qinghua Li

Thesis Committee:

Signature:  Date: 4/12/2021

Printed: Matthew Patitz

Signature:  Date: 4/15/2021

Printed: Dong Jin

***Abstract***—Automatic Generation Control (AGC) is a key control system utilized in electric power systems. AGC uses frequency and tie-line power flow measurements to determine the Area Control Error (ACE). ACE is then used by the AGC to adjust power generation and maintain an acceptable power system frequency. Attackers might inject false frequency and/or tie-line power flow measurements to mislead AGC into falsely adjusting power generation, which can harm power system operations. Various data forgery detection models are studied in this thesis. First, to make the use of predictive detection models easier for users, we propose a method for automated generation of detection threshold for Long-Short-Term-Memory neural network based detection models. Second, we study the performance of various detection models under low-rate false data injection attacks.

## I. INTRODUCTION

Automatic Generation Control (AGC) is a key control system in the power grid that aims to maintain an acceptable power system frequency (i.e. 60 Hz in the U.S.) for each control area. A diagram of the AGC system is shown in Fig. 1. AGC periodically calculates the Area Control Error (ACE) based on the frequency and tie-line power flow measurements (i.e., the amount of power exchange with neighboring control areas). AGC will use the ACE value to automatically adjust power generation in order to maintain a stable power system frequency. However, attackers might inject false frequency and tie-line power flow measurements to mislead AGC to calculate a false ACE value. AGC will then falsely adjust power generation based on the false ACE value. This false adjustment causes harm to the power grid operations. [1]

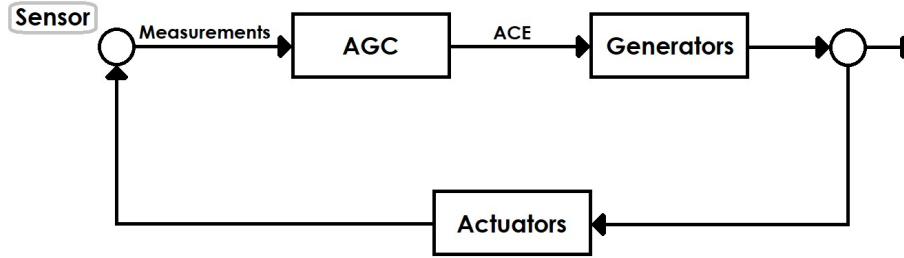


Figure 1: Automatic Generation Control (AGC) System

There exist various models to detect data forgery in AGC. Predictive models use a subset of the previous data to predict the next data. The predictions are then compared to the observed data in order to identify attacks. Signal processing models can also detect data forgery with great accuracy. For example, Fourier transform based attack detection models [1] use rolling averages and frequencies to identify if a subset of the data has been manipulated. In section III, we will discuss the following data forgery detection models in more detail: long short-term memory (LSTM) neural networks, random forest, and Fourier transform. The first two are machine learning-based predictive models, and the third is a signal processing model.

Predictive detection models require a threshold value in order to detect attacks by comparing predicted and real data and assessing their difference. The optimal detection threshold depends on multiple factors, like standard deviation, dataset size, etc. It is usually manually tuned which takes much time. To simplify the process, in this thesis, we develop a method to automatically generate a threshold value for the LSTM based models. The details are discussed in section IV.

When false data is injected suddenly and/or in a high magnitude, models can more easily detect the attack. This is because there will be a sharp increase or decrease in the data, which means that the absolute difference between predicted data and observed data will be larger than normal, and similarly for Fourier transform, the rolling average will sharply increase or decrease more than

normal. However, specially crafted low-rate attacks might not be detected by the models, because the attacks start out very small and slowly increase. In this thesis, we study low-rate ramp attacks and evaluate various detection models' performance under such attacks. A low-rate ramp attack is an attack that slowly changes data values (the rate of change is governed by a ramp parameter) but is run over an extended period of time. Sections V and VI provide more details about the basic ramp attacks and the low-rate ramp attacks.

## II. RELATED WORK

There has been work done on detecting attacks in AGC [1, 3, 4, 5, 6]. However, none of them study low-rate attacks as this thesis does. The closest work [2] discusses low-rate ramp attacks against predictive filters in phasor measurement units (PMUs). [2] compares a low-rate ramp attack to a prolonged sudden increase attack (adds a constant large amount to the real data for a variable amount of time). In this thesis, we compare low-rate ramp attacks to a quicker ramp attack that uses a comparatively larger parameter and much shorter attack length. [2] uses two moving linear regression formulas (two single-feature linear predictive methods) to predict values for attack detection. In this thesis, we study machine learning and signal processing models to detect attacks. In addition, this thesis also discusses the relationship between threshold values and ramp attack detection. Whereas [2] does not use or discuss any threshold values to detect attacks; it displays the absolute difference between predictions and real data, called "observation residues" in [2].

## III. DETECTION MODELS

We split up the data forgery detection models into two groups: predictive and signal processing. The predictive detection models consist of the following: multi feature LSTM, single feature LSTM, multi feature random forest, and single feature random forest. The signal processing detection models consist of the following: Fourier transform min based detection and Fourier transform sum based detection [1].

Single feature LSTM and single feature random forest detection models [1] use the ACE data for detection. They only use one input data feature, so they are called single feature models. After a model is properly trained, the model predicts the ACE value for the next cycle using the previous 5 cycles of real ACE data. Similarly, the model will predict the data for every cycle of the remaining data. To detect attacks, the model will look at each 8 cycles of observed ACE data as well as the predictions for those cycles. The model will calculate the sum of absolute differences between the predictions and observed data for the 8 cycles. If the sum of absolute differences is larger than the detection threshold, those 8 cycles are marked as potentially attacked. The only difference between the LSTM and random forest detection models is the neural network architecture.

Multi feature LSTM and multi feature random forest detection models take in the following input data features: ACE, frequency, tie-line power flow, and optionally load and/or load forecast. They are capable of utilizing multiple input data, so they are called multi feature models. For this thesis, we will focus on using all 5 input data, because that has the best performance. The prediction method is very similar to the single feature models. The only change is that multi feature models will use the previous 5 cycles of each inputted data to predict the next cycle of each inputted data. The detection method is the exact same as the single feature models.

Fourier transform sum and Fourier transform min based detection models use the ACE data for detection. The models first calculate a rolling average for every 10 cycles. The rolling averages are then converted to frequencies with Fast Fourier Transform (fft). For min based detection, we store the minimum frequency value for each 10 frequencies. For sum based detection, we store the sum of the absolute value for each 10 frequencies. To detect attacks, min based detection compares each of the stored minimum frequencies to the detection threshold; if any of the minimum frequencies are less than the threshold, the 10 cycles that frequency represents are marked as attacked. To detect attacks, sum based detection compares each of the stored sums of frequencies to the detection threshold; if any of the sums are larger than the threshold, the 10 cycles that sum represents is marked as attacked.

#### IV. AUTOMATIC THRESHOLD GENERATION FOR LSTM

The ideal threshold depends on a lot of factors in the LSTM-based detection method. For example, we would use a relatively small threshold for a larger dataset; we would use a relatively large threshold for a smaller dataset. This is because the LSTM model predictions will become more accurate the larger the training data size, and as a result we can use a lower threshold. Additionally, the larger the standard deviation in a dataset, the larger our detection threshold should be. This is because there will likely be a larger gap between predicted and observed data for a larger standard deviation. Thresholds are usually manually tried and determined, which takes much time and has high requirements in the user's technical background.

To solve this problem, we develop a method to automatically generate a detection threshold. The program will start by first training the LSTM model. With the trained model, it then generates predictions on the training dataset. Thus, it is predicting values on the same dataset that was used



to train the LSTM model. The program will then calculate the absolute difference between the predicted and observed data values. For every possible series of 8 clock cycles (8 data values that are right next to each other), we calculate the sum of the absolute differences for each of those 8 cycle series. The reason we used 8 clock cycles is because according to [1], the shortest data forgery attack that can still cause damage to the power grid is 10 cycles. Therefore, we would like to choose a number of cycles less than 10, in order to detect the attacks before they inflict damage. Now the program has a large list of absolute differences for 8 cycle series. In this list, a majority of the list will likely be very small values. This is because the predictions on the training dataset will be very accurate. However, there will still be values that are relatively larger; this is because of LSTM's forget gate and general prediction faults. The forget gate is something that LSTM has in its neural network architecture. LSTM's forget gate will let the LSTM model not learn from certain parts of the training dataset. General prediction faults just reference the fact that a predictive model will not be 100% accurate. From this list of absolute differences, the program will then select a specified percentile as our generated detection threshold. We will refer to this percentile as the *generation percentile*. If we chose 100 as our generation percentile, the program would choose the largest value from the list of absolute differences. If we had chosen 0 as our generation percentile, the program would choose the smallest value from the list.

In order to test automatic threshold generation for LSTM, we took the results of multiple generation percentiles' performance on random, scale, and ramp attacks for single-feature LSTM. For multi-feature LSTM, we took the performances on random, scale, ramp, min, and max attacks. A ramp attack is defined in Fig. 3. A random attack adds a random amount, within a specified range, to the real data. A scale attack takes the sum of the real data and the real data multiplied by a specified scale parameter. Min and max attacks replace the real data with the minimum or

maximum value seen in the previous cycles. We then averaged all of collected true and false positive detection accuracies for each model. For multi-feature LSTM we also averaged the attack localization accuracies. Attack localization is when the model determines which feature the data forgery attack originated from, tie-line power flow or frequency. The results are shown in Fig. 2.

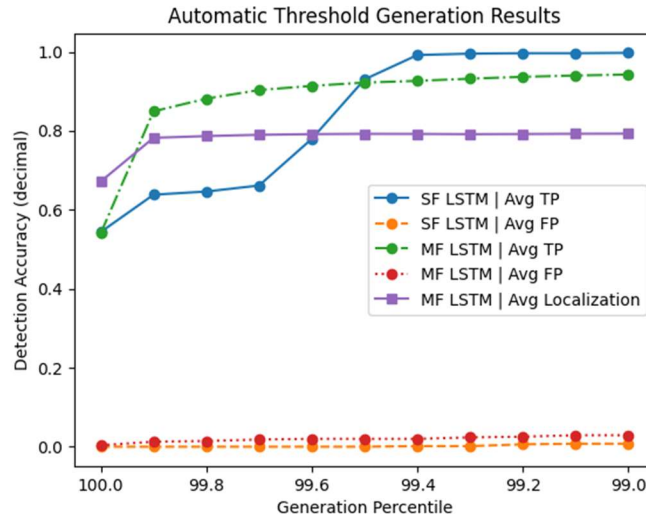


Figure 2: Automatic Threshold Generation Results

*SF = single feature, MF = multi feature, TP = true positive, FP = false positive*

As shown in Fig. 2, choosing 100 as the generation percentile causes a relatively low true positive (a true positive is a case where a true attack is detected as an attack) accuracy for both LSTM models. This is because the program will choose the largest potential threshold value. However, choosing a generation percentile closer to 99 has quite good performance. We can see that both the single feature and multi feature LSTM models have above 96% average true positive while still having less than a 5% average false positive (a false positive is a case where a normal event is detected as an attack).

To perform the tests shown in Fig. 2, we used a 50k cycle subset of a simulated AGC dataset containing more than 600k cycles. This 600k cycle AGC dataset is the same as the dataset used for other tests performed for this thesis.

## V. RAMP ATTACK ANALYSIS

Ramp attacks require a ramp parameter,  $\lambda$ . Let  $t_0$  represent the cycle when an attack starts. Let  $f(t)$  represent the data as a function of cycles. A ramp attack could be defined as such (Fig. 3):

$$f(t) = \begin{cases} f(t), & t \notin \text{Attack} \\ f(t) + \lambda \cdot (t - t_0 + 1), & t \in \text{Attack} \end{cases}$$

Figure 3: Ramp Attack Function

Fig. 4 shows the detection accuracies for each detection model on ramp attacks with various ramp parameters,  $\lambda$ , and an attack length of 10 cycles. These tests were done on a 100k cycle subset of a simulated AGC dataset containing more than 600k cycles. This subset is split into 2 parts by the predictive models: training and testing dataset. For the testing results below, the training dataset is the first 67% of the data, and the testing dataset is the remaining 33% of the data. The training dataset is used to train the predictive model, and the testing dataset is what the detection results are based on. The unattacked ACE testing data is shown in Fig. 5. The signal processing models provide detection results over the entire data, training and testing datasets. The ramp attacks were injected periodically throughout the testing dataset for predictive models, and periodically throughout the entire dataset for the signal processing models.

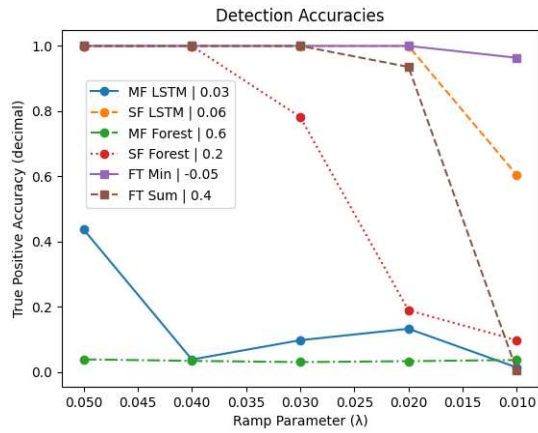


Figure 4: True Positive Detection Accuracies

*Legend Format: model name | threshold*

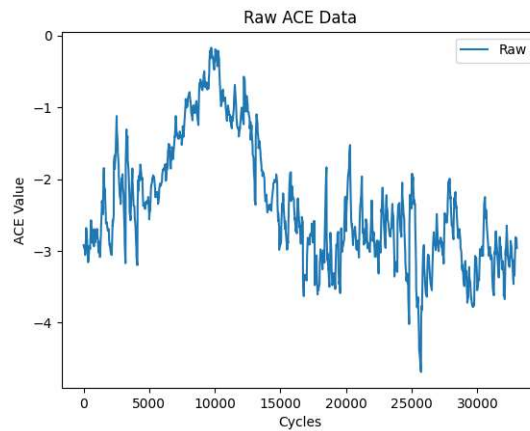


Figure 5: Raw ACE Data

Excluding multi feature models, the detection models only use the ACE data to perform any attack detection. In single feature predictive models, we assume that predictions are nearly exact to the real data. It will be theoretically impossible to detect ramp attacks when the detection threshold is greater than the ramp attack parameter multiplied by 8 (threshold  $> \lambda \cdot 8$ ). Other single feature predictive models can replace “ $\lambda \cdot 8$ ” with “ $\lambda \cdot$ [the number of cycles the model looks at to

compare to the threshold]”. In the tested single feature predictive models, they use 8 cycles of data to compare to the threshold. For example, let us consider a detection threshold of 0.1 and a ramp attack parameter,  $\lambda$ , of 0.01. When the ramp attack first starts,  $t_0$ , there will be around a 0.01 difference between the predicted and observed ACE value, because the model will be basing the prediction on the previous 5 cycles, which were not attacked. The prediction for the next cycle,  $t_1$ , will be based on the previously attacked cycle as well as 4 not attacked cycles. Due to  $t_0$  being directly prior to  $t_1$ ,  $t_0$  carries a more weight than the 4 not attacked cycles in how the model predicts  $t_1$ . As a result, we can assume that the model will again have around a 0.01 difference between the predicted and observed data. Similarly, as the model continues to predict the data, the predictions become closer and closer to the attacked data values, because the previous cycles that the model looks at to make predictions will also be attacked. Therefore, at the best case, there will be a sum of absolute differences of around 0.08 for 8 cycles; however, it is more likely to be less than 0.08. This attack will not be detected by the data forgery model, because 0.08 is not greater than our suggested detection threshold, 0.1. If predictions were nearly exact to the real data, a solution would be to simply make the detection threshold as small as needed. However, predictions are not nearly exact to the real data a majority of the time, because ACE data is constantly fluctuating in unique patterns; therefore, reducing the threshold will increase false positives and will not be a viable solution in most cases. Similarly, the Fourier transform based detection models will also struggle to detect ramp attacks when the ramp parameter is lower. This is because Fourier transform models detect attacks based on the larger fluctuations in the data, and if the ramp parameter is relatively small compared to the detection threshold, the fluctuations in the data will not be detected. Multi feature models are a bit more complex, because they are able to utilize load and load forecast data to make predictions. However, they are still predictive models and are

susceptible to ramp attacks with smaller ramp parameters, in a similar fashion to single feature predictive models.

If we see the results for the predictive models, Fig. 4, we notice that the LSTM models have a better performance than the random forest models. This is because LSTM is more suited for data forgery detection in AGC. LSTM uses patterns and trends in the training dataset to make predictions in the testing dataset. Random forest uses an average of predictions from multiple decision trees (bagging) created from the training dataset. Random forest is also not capable of extrapolating, whereas LSTM can extrapolate. Extrapolating means to return a prediction value outside of the range given in the training dataset. Random forest is also not as good as LSTM in predicting versus time; this is because random forests are not as good at predicting on patterns and trends. Random forests also do not work that well given multiple features/categories, especially when one of the features are more important than the rest. In our case, the ACE data is more important than the rest. When a random forest is given multiple features, it will make random decisions on which features to exclude at nodes. This can create a prediction bias, and even when it is averaged with other trees, the bias will remain. This will make the predictions seem more “average” many times. This is also a reason why multi feature random forest will not be as good vs time/trends. This is also a reason why single feature random forest appears to perform better than multi feature random forest.

## VI. LOW-RATE RAMP ATTACK

The basic idea behind low-rate ramp attacks is to perform a ramp attack, with a very small ramp parameter, for an extended period of cycles. The dataset used to test the low-rate ramp attacks is the same dataset used in section V. The ACE values of the testing dataset are shown in Fig. 5.

The ramp attacks in section V each had attack lengths of 10 cycles and were injected periodically throughout the testing dataset for predictive models and periodically throughout the entire dataset for the signal processing models. The low-rate ramp attacks will be injected across the entirety of the testing dataset only. The results for the raw dataset and all tested low-rate ramp attacks are shown in Fig. 6.

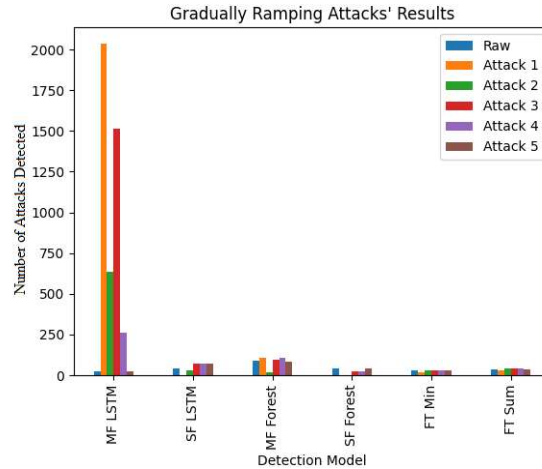


Figure 6: Low-rate Ramp Attacks' Detection Results

First, we conducted tests on the raw dataset, not attacked. This will let us know around how many false positives we can expect. The raw ACE testing data is shown in Fig. 5.

The first low-rate ramp attack is defined below (Fig. 7):

$$f(t) = \begin{cases} f(t), & t \notin \text{Attack} \\ f(t-1) + \lambda, & t \in \text{Attack} \end{cases}$$

Figure 7: First Attack Function

In the first attack function, the ramp parameter,  $\lambda$ , is set to -0.00001. The function is only applied to the ACE data. The purpose of the first low-rate ramp attack is to see how each detection model

performs against a single feature pattern. A comparison between the raw and first low-rate ramp attack ACE testing data is shown below in Fig. 8.

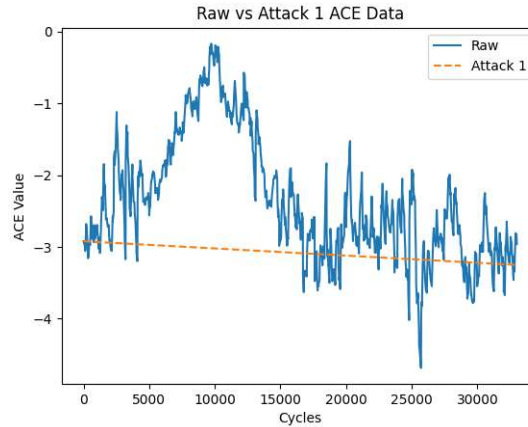


Figure 8: Raw vs Attack 1 ACE Data

As shown in Fig. 6, the first low-rate ramp attack was only detected by multi feature LSTM. The rest of the models did not detect the attack because the number of attacks detected are similar or lower than the general false positives of the raw data. We assume that multi feature models are capable of detecting this attack due to the weight the other features have on predictions.

The second low-rate ramp attack uses the same function as in attack 1 and defined in Fig. 7. However, the function is only applied to the tie-line power flow data. The changes in tie-line power flow are reflected in the ACE values by using the following function shown in [1]:

$$ACE = (P_{tie\_line} - P_{sch}) + B(f - f_{sch})$$

In the function above, “ $P_{sch}$  and  $f_{sch}$  are the scheduled tie-line power flow and the scheduled frequency respectively.  $P_{tie\_line}$  and  $f$  are measurements.  $B$  is the frequency bias factor, which is constant for each power system and is estimated annually” [1]. The purpose of the second low-rate ramp attack is to see if there will be a change in detection if the attack originates from the tie-line



power flow. A comparison between the raw and second low-rate ramp attack ACE testing data is shown below in Fig. 9.

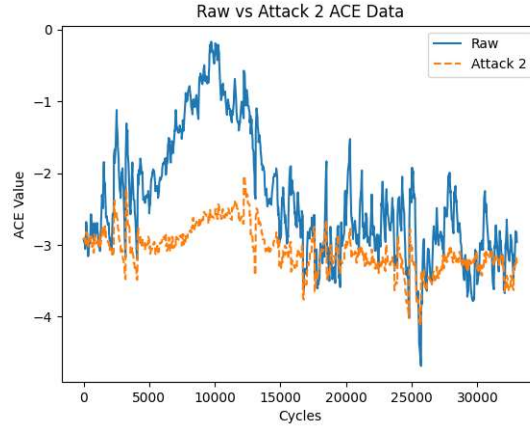


Figure 9: Raw vs Attack 2 ACE Data

As shown in Fig. 6, the second low-rate ramp attack was also only detected by multi feature LSTM. We assume that multi feature models are capable of detecting this attack due to the weight the load and load forecast data have on predictions. We did test the second attack on multi feature LSTM without including the load and/or the load forecast data, and the attack was not detected.

The third low-rate ramp attack uses the normal ramp attack function defined in Fig. 3. The ramp parameter,  $\lambda$ , is set to -0.00001. Similar to the first low-rate ramp attack, the function is only applied to the ACE data. The purpose of the third low-rate ramp attack is to test the detection models performance on a more traditional low-rate ramp attack. A comparison between the raw and third low-rate ramp attack ACE testing data is shown below in Fig. 10.

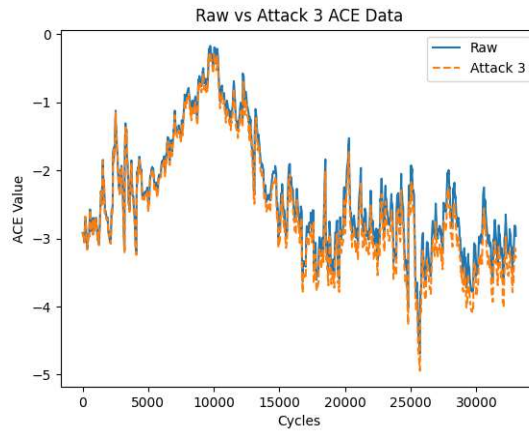


Figure 10: Raw vs Attack 3 ACE Data

As shown in Fig. 6, the third low-rate ramp attack was also only detected by multi feature LSTM. We assume that multi feature models are capable of detecting this attack due to the weight the other features have on predictions. More specifically, as more cycles pass, the absolute difference between the predicted values and real values increases, because the attack is ramping the ACE values and not the other features' values.

The fourth low-rate ramp attack uses the same function as in attack 3 and defined in Fig. 3. However, the function is only applied to the tie-line power flow data. The changes in tie-line power flow are reflected in the ACE values with the same method as in attack 2. The purpose of the fourth low-rate ramp attack is to see if there will be a change in detection if the attack originates from the tie-line power flow. A comparison between the raw and fourth low-rate ramp attack ACE testing data is shown below in Fig. 11.

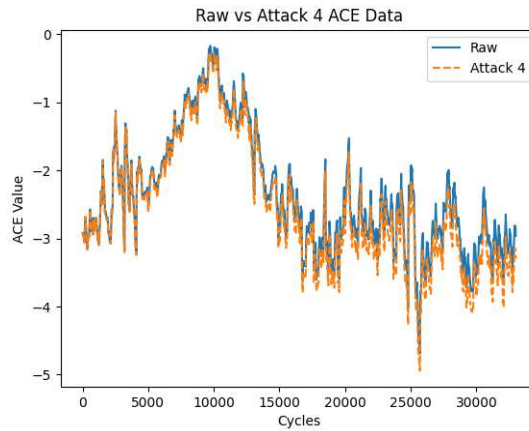


Figure 11: Raw vs Attack 4 ACE Data

As shown in Fig. 6, the fourth low-rate ramp attack was also only detected by multi feature LSTM. However, the attack was only barely detected. We assume that multi feature models are capable of detecting this attack due to the weight the load and load forecast data have on predictions. More specifically, as more cycles pass the absolute difference between the predicted values and real values increases, because the attack is ramping the ACE and tie-line power flow values and not the load and load forecast values. However, the weight that the load and load forecast have on predictions seems to be much smaller than the other features, because there are a relatively smaller number of attacks detected. We did test the fourth attack on multi feature LSTM without including the load and/or the load forecast data, and the attack was not detected.

The fifth low-rate ramp attack is the same as the fourth low-rate ramp attack, except the ramp parameter,  $\lambda$ , is set to -0.000001. The purpose of the fifth attack is to check if the attack will be detected by any model if the ramp parameter was lowered even further than the fourth low-rate ramp attack. A comparison between the raw and fifth low-rate ramp attack ACE testing data is shown below in Fig. 12. A closer look at the last few cycles is also shown below in Fig. 13.

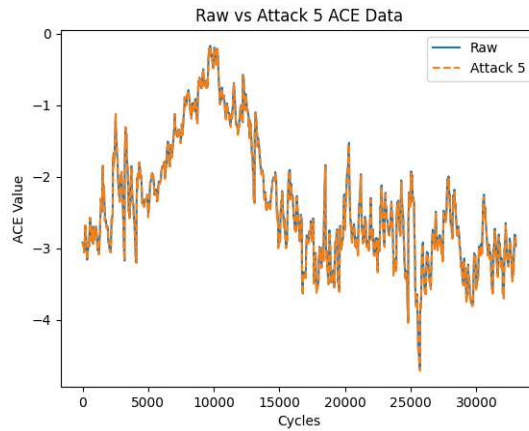


Figure 12: Raw vs Attack 5 ACE Data

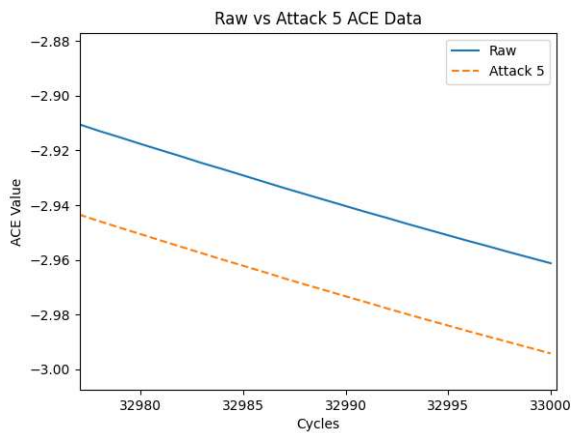


Figure 13: Raw vs Attack 5 ACE Data - Closer Look

As shown in Fig. 6, the fifth low-rate ramp attack was not detected by any data forgery detection models.

As shown in Fig. 6, multi feature LSTM was the only data forgery detection model to detect any of the low-rate ramp attacks. This is because the single feature predictive models and the signal processing models struggle against attacks with a small ramp parameter as analyzed in section V.

Multi feature random forest also struggles to perform well due to the reasons described in section V. Multi feature LSTM also has the advantage of having multiple features to utilize for predictions.

Multi feature predictive models might be capable of detecting these attacks better, if the load and load forecast had more weight on the predictions, because those values are not tampered with or affected when injecting false data into tie-line power flow or frequency. This means that if they had more weight on predictions, the absolute difference between the predicted and observed values could be larger, therefore resulting in more detections. However, adding more weight to load and load forecast data could also cause predictions to be less accurate, and this would result in more false positives as well as true positives.

## VII. CONCLUSION

This thesis proposed a method for automated threshold generation for LSTM models and evaluated its performance. Given an appropriate generation percentile, our automated threshold generation method is very effective as shown in Fig. 2. We analyzed and showed results of traditional ramp attacks in section V. We explored 5 different low-rate ramp attacks and showed their results against the data forgery detection models in section VI. We found that well crafted low-rate ramp false data injection attacks are able to avoid detection, as shown by the fifth attack in section VI and Fig. 6.

In the future, more work can be done regarding automated threshold generation for detection models other than LSTM. This method works well for LSTM, due to LSTM's forget gate. Additionally, a more universal automatic threshold generation method could be explored.

More exploration could also be done on the detection of low-rate ramp attacks with the use of multi feature predictive data forgery detection models.

## ACKNOWLEDGMENT

This material is based upon work supported by the Department of Energy under Award Number DE- OE0000779.

## REFERENCES

- [1] Fengli Zhang and Qinghua Li. "Deep Learning-Based Data Forgery Detection in Automatic Generation Control." In 2017 IEEE Conference on Communications and Network Security (CNS): International Workshop on Cyber-Physical Systems Security (CPS-Sec).
- [2] Zhigang Chu, Andrea Pinceti, Reetam Sen Biswas, Oliver Kosut, Anamitra Pal, and Lalitha Sankar. "Can Predictive Filters Detect Gradually Ramping False Data Injection Attacks Against PMUs?," 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Beijing, China, 2019, pp. 1-6, doi: 10.1109/SmartGridComm.2019.8909739.
- [3] Tan, Rui, Hoang Hai Nguyen, and Hoay Beng Gooi. "Optimal False Data Injection Attack against Automatic Generation Control in Power Grids." In 2016 ACM/IEEE 7<sup>th</sup> International Conference on Cyber-Physical Systems (ICCPS), pp. 1-10. IEEE, 2016.
- [4] Ali, Muhammad Qasim, et al. "Two-tier data-driven intrusion detection for automatic generation control in smart grid." Communications and Network Security (CNS), 2014 IEEE Conference on. IEEE, 2014.

- [5] S. Sridhar, and M. Govindarasu. "Model-based attack detection and mitigation for automatic generation control." *Smart Grid, IEEE Transactions on* 5.2 (2014): 580-591.
- [6] Vrakopoulou, Maria, et al. "Cyber-attacks in the automatic generation control." *Cyber Physical Systems Approach to Smart Electric Power Grid*. Springer Berlin Heidelberg, 2015. 303-328.