Data Correction Library (PMUDataCorrector) User Manual

Version 1.1.1

Dilan Senaratne, Travis Hagan, Rich Meier, Eduardo Cotilla-Sanchez, Jinsub Kim

September 2, 2022

Contents

1 Introduction

This document details how to implement the Data Correction Library (PMU-DataCorrector) on a SEL Real-Time Automation Controller (RTAC).

The PMUDataCorrector is used in combination with user provided files to detect and correct wide-area synchrophasor data that has been corrupted by a GPS spoofing attack. GPS (Global Position System) spoofing attacks are relatively easy attacks to propagate requiring only a few hundred dollars worth of readily available equipment. During an attack, the angle of synchrophasor data is affected because the time at the substation has been modified, thereby providing data that is out of sync from the rest of the system.

This may result in undesired alarms, worsening operating conditions, misoperation of protection devices, and general confusion by operators. The PMUDataCorrector described here can detect and correct angle data during sophisticated spoofing attacks. It has been tested on IEEE RTS1996 test case with up to **four** simultaneous attacked substations.

First, the basics of the system are described in the introduction. Second, the necessary data pre-processing is explained in section 2. Third, the theory and inner workings of the PMUDataCorrector are described in section 3. Fourth, a complete working system using synchrophasor data is detailed in section 4. Lastly, examples of the system under various operating conditions are provided in section 5.

1.1 PMU measurement model

Phasor Measurement Units (PMU) are grid connected devices specifically designed to enhance wide-area monitoring by creating accurate time-synchronized synchrophasors and streaming that data to a PDC (Phasor Data Concentrator) or control room for analysis and storage. PMUs are generally configured with a high data rate of 30 to 60 samples per second. Synchrophasors are not possible under Supervisory Control And Data Acquisition (SCADA) systems because SCADA lacks a precise timing element to synchronize data and the bandwidth to transmit the data. SCADA also reports at a much slower rate, of around one sample per second. The rest of this section details how PMU generated synchrophasor data is used in the PMUDataCorrector.

The power network is represented by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where the vertices ($\mathcal{V} = 1, 2, \dots, N$) represent the set of N buses and the edges ($\mathcal{E} = 1, 2 \dots, M$) represent the set of M branches. Let $x_i^{(t)}$ denote the voltage phasor at bus i and time t. If a PMU is placed at bus i then it can measure the bus voltage phasor at time t which is denoted by $z_{V_i}^{(t)}$. Thus,

$$z_{V_i}^{(t)} = x_i^{(t)} \quad t = 1, 2, ... \tag{1}$$

PMUs can also be placed and configured to read/record current that enters or leaves a bus/substation via its adjacent/interconnected branches. Let us define the set of all remote buses that are interconnected to bus i via a branch as well as measured by a PMU placed at bus i as $\mathcal{M}_i$. Therefore a particular remote bus $l \in \mathcal{M}_i$. The current phasor measured at time t by a PMU placed at bus i between buses i and l is therefore denoted: $z_{I_{il}}^{(t)}$. Mathematically, the current phasor can be calculated using the following equation:

$$z_{I_{il}}^{(t)} = y_{il}(x_i^{(t)} - x_l^{(t)}) + j\frac{b_{il}^s}{2}x_{il}^{(t)} \quad t = 1, 2, ... \tag{2}$$

where y_{il} is the series admittance of the branch $\{i, l\}$ and b_{il}^s is the line charging susceptance.

Let $\mathbf{x}^{(t)} = [x_1^{(t)}, x_2^{(t)}, \dots, x_N^{(t)}]$ denote the voltage phasors of the total N buses at time t. From equation (1) and (2) we can see that the measurement at time t has a linear relationship to the bus voltage phasors and branch current (i.e., system state) at time t as follows.

$$\mathbf{z}^{(t)} = \mathbf{H}\mathbf{x}^{(t)} + \mathbf{e}^{(t)} \quad t = 1, 2, ... \tag{3}$$

where $\mathbf{z}^{(t)}$ is the augmented PMU measurements for each PMU at time t. The vector $\mathbf{z}^{(t)}$ is constructed by concatenating the voltage phasor and all the measured current for each PMU together (i.e, $\mathbf{z}^{(t)} = [z_{V_i}^{(t)}, z_{I_{il}}^{(t)}]^T \; \forall \; i$ and l). The linear operator determined by system topology and line parameters is denoted by $\mathbf{H}$ and the complex Gaussian noise is denoted by $\mathbf{e}^{(t)}$.

1.2 GPS spoofing attack

PMUs make use of a precise time reference to compute phase angle measurements. A GPS spoofing attack may shift this time reference resulting in a common bias to all the phase angle measurements collected by the subset of PMUs that have been spoofed. PMus are also installed only on a subset of buses. Let $\mathcal{T}$ denote the set of buses with PMU installed (i.e., $\mathcal{T} \subseteq \mathcal{V}$). The total number of PMUs is denoted by $|\mathcal{T}| = K$. Let $\alpha_k^{(t)}$ denote the angle bias introduced by the spoofing attack on all of the phase angle measurements of PMU k, located at bus i, occurring at time t. Given this angle bias, the spoofed voltage and current measurements at time t, denoted by $\bar{z}_{V_i}^{(t)}$ and $\bar{z}_{I_{il}}^{(t)}$ respectively, are defined as follows:

$$\bar{z}_{V_i}^{(t)} = e^{j\alpha_k^{(t)}} z_{V_i}^{(t)} \tag{4}$$

$$\bar{z}_{I_{il}}^{(t)} = e^{j\alpha_k^{(t)}} z_{I_{il}}^{(t)} \tag{5}$$

Given the above definitions, we can define the time-domain vector of spoofed and unspoofed PMU voltage and current measurements as: $\bar{\mathbf{z}}_k^{(t)} = [\bar{z}_{V_i}^{(t)} \; \bar{z}_{I_{il}}^{(t)}]$ and $\mathbf{z}_k^{(t)} = [z_{V_i}^{(t)} \; z_{I_{il}}^{(t)}]$ for all $l \in \mathcal{M}_i$. This notation can be simplified for all the measurements of PMU k and written as,

$$\bar{\mathbf{z}}_k^{(t)} = e^{j\alpha_k^{(t)}} \mathbf{I}_{m_k} \mathbf{z}_k^{(t)} \tag{6}$$

where $\mathbf{I}_{m_k}$ is an identity matrix with size $m_k \times m_k$ and m_k is the number of measurements associate with PMU k. This can be generalized for the entire PMU network as follows,

$$\begin{bmatrix} \bar{\mathbf{z}}_1^{(t)} \\ \bar{\mathbf{z}}_2^{(t)} \\ \vdots \\ \bar{\mathbf{z}}_K^{(t)} \end{bmatrix} = \begin{bmatrix} e^{j\alpha_1^{(t)}}\mathbf{I}_{m_1} & 0 & \dots & 0 \\ 0 & e^{j\alpha_2^{(t)}}\mathbf{I}_{m_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & \dots e^{j\alpha_K^{(t)}}\mathbf{I}_{m_K} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1^{(t)} \\ \mathbf{z}_2^{(t)} \\ \vdots \\ \mathbf{z}_K^{(t)} \end{bmatrix}$$

$$\bar{\mathbf{z}}^{(t)} = \phi(\boldsymbol{\alpha}^{(t)})\mathbf{z}^{(t)} \tag{7}$$

$$\bar{\mathbf{z}}^{(t)} = \phi(\boldsymbol{\alpha}^{(t)})(\mathbf{H}\mathbf{x}^{(t)} + \mathbf{e}^{(t)}) \quad t = 1, 2, \dots \tag{8}$$

where $\phi(\boldsymbol{\alpha}^{(t)})$ denotes the attack structure and $\boldsymbol{\alpha}^{(t)} = [\alpha_1^{(t)}, \alpha_2^{(t)}, \dots, \alpha_K^{(t)}]^T$. The $\alpha_k^{(t)}$ is nonzero only if there is an attack at PMU k at time t. Thus the corresponding diagonal block of $\phi(\boldsymbol{\alpha}^{(t)})$ is also non zero. If there is no attack at PMU k at time t then the corresponding diagonal block of $\phi(\boldsymbol{\alpha}^{(t)})$ is an identity matrix.

1.3 Algorithm

Based on the PMU placement and the topology, the power network can be decomposed into several zones. For more information regarding this decomposition please refer [1]. One important observation in the measurement model (1) and (2) after decomposing the network into several zones is that the measurement for each zone depends only on the state variable of that zone alone. Let γ denote the zone and there is Γ number of zones. Then the PMU spoofed measurement model (8) has the following block structure.

$$\begin{bmatrix} \bar{\mathbf{z}}^{(t),1} \\ \vdots \\ \bar{\mathbf{z}}^{(t),\Gamma} \end{bmatrix} = \begin{bmatrix} \phi_1(\alpha^{(t),1}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \phi_\Gamma(\alpha^{(t),\Gamma}) \end{bmatrix} \begin{bmatrix} \mathbf{H}^1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathbf{H}^\Gamma \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(t),1} \\ \vdots \\ \mathbf{x}^{(t),\Gamma} \end{bmatrix} + \begin{bmatrix} \mathbf{e}^{(t),1} \\ \vdots \\ \mathbf{e}^{(t),\Gamma} \end{bmatrix}$$

where $\bar{\mathbf{z}}^{(t),\gamma}$ is the sub vector of spoofed measurement vector $\bar{\mathbf{z}}^{(t)}$ for zone γ, $\bar{\mathbf{x}}^{(t),\gamma}$ is the sub vector of the state vector $\bar{\mathbf{x}}^{(t)}$ for zone γ, $\bar{\mathbf{e}}^{(t),\gamma}$ is the sub vector of the noise vector $\bar{\mathbf{e}}^{(t)}$ for zone γ, $\mathbf{H}^1$ is the sub matrix of $\mathbf{H}$, $\alpha^{(t),\gamma}$ denotes the angle biased introduced to PMUs in zone γ and $\phi_\gamma(\alpha^{(t),\gamma})$ is a diagonal matrix which denotes a sub matrix of $\phi(\alpha^{(t)})$ for zone γ. Based on this notation, we can represent the spoofed measurement model for zone γ as follows.

$$\bar{\mathbf{z}}^{(t),\gamma} = \phi_\gamma(\alpha^{(t),\gamma})\mathbf{H}^\gamma\mathbf{x}^{(t),\gamma} + \mathbf{e}^{(t),\gamma} \quad t = 1, 2, ... \tag{9}$$

The decomposed spoofed measurement model in equation (9) can be rewritten as follows,

$$\phi_\gamma^{-1}(\alpha^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma} = \mathbf{H}^\gamma\mathbf{x}^{(t),\gamma} + \mathbf{e}^{(t),\gamma} \quad t = 1, 2, ... \tag{10}$$

The above equation implies that if $\alpha^{(t),\gamma}$ contain correct angle biases, then $\phi_\gamma^{-1}(\alpha^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma}$ would reside very close to the column space of $\mathbf{H}^\gamma$. Thus, if we project $\phi_\gamma^{-1}(\alpha^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma}$ to the $R(\mathbf{H}^\gamma)$, where $R(\mathbf{H}^\gamma)$ denotes the range space of $\mathbf{H}^\gamma$ then the projection residue $\mathbf{r}^\gamma$ can be given as,

$$\mathbf{r}^\gamma = (\mathbf{I}_{m^\gamma} - \mathbf{P}_{H^\gamma})\phi_\gamma^{-1}(\alpha^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma} \quad t = 1, 2, ... \tag{11}$$

where $\mathbf{P}_{H^\gamma}$ denotes the projection operator for projection onto the column space of $\mathbf{H}^\gamma$ (i.e., $\mathbf{H}^\gamma(\mathbf{H}^{\gamma^T}\mathbf{H}^\gamma)^{-1}\mathbf{H}^{\gamma^T}$). We refer the matrix $(\mathbf{I}_{m^\gamma} - \mathbf{P}_{H^\gamma})$ orthogonal projection or modal matrix. We propose the following optimization problem to find the most sparse $\alpha^{(t),\gamma}$ that makes the projection residue no greater than the predefined threshold τ^γ as follows,

$$\begin{aligned} \hat{\alpha}^{(t),\gamma} = \underset{\alpha^{(t),\gamma}}{\text{argmin}}\ \|\alpha^{(t),\gamma}\|_0 \quad t = 1, 2, ... \\ \text{subject to} \|(\mathbf{I}_{m^\gamma} - \mathbf{P}_{H^\gamma})\phi_\gamma^{-1}(\alpha^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma}\|_2^2 \leq \tau^\gamma \end{aligned} \tag{12}$$

2 Data pre-processing

The library required power system network information such as topology, line parameters and PMU placement to perform the data correction algorithm.

The first step is to preprocess all the network information and send them to the PMUDataCorrector library. To do that a python script is provided along with some sample input .csv files to the script. The user must first run the given python script and obtain the necessary output .csv files. Figure 1 shows overall procedure to prepare the input files to the RTAC library.

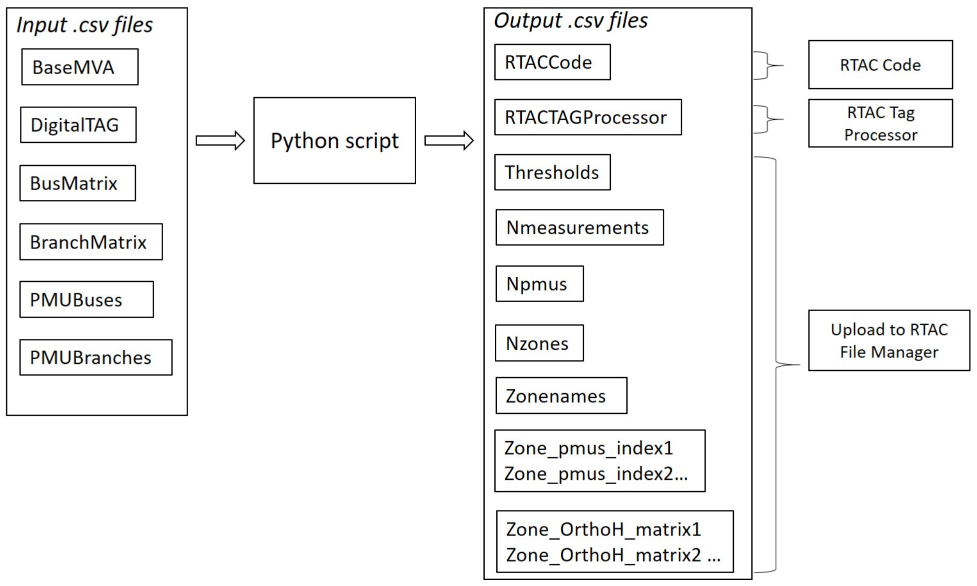


Figure 1: Procedure to generate RTAC input

2.1 Input files

The user should save the following ".csv" files to the same folder where the script file is located.

- **BaseMVA** : The system MVA base used for converting power into per unit quantities
- **DigitalTAG** : The tag name in RTAC that used to send Boolean output of the presence of the attack
- **BusMatrix** : The network bus information
- **BranchMatrix** : The network branch information
- **PMUBuses** : PMU locations
- **PMUBranches** : Branches that measured by PMUs

BusMatrix and BranchMatrix are the output of the psse2mpc function of MATPOWER [2]. The user can copy and paste the mpc.bus of the output to BusMatrix.csv and mpc.branch to BranchMatrix.csv. The column names of these matrices should match the column names given in the MATPOWER user manual [1] [3]. The user can use the provided sample files to create new files.

The first column in PMUBuses.csv consists of the bus indices that contain a PMU. The second column consists of the corresponding voltage phasor TAG names of the RTAC C37.118 client devices. The third column consists of the phasor TAG names of the RTAC C37.118 server devices which are used to send the corrected voltage phasors. The fourth column consists of the analog output TAG names of the RTAC C37.118 server devices which are used to output the estimated angle biases. The fifth column and beyond contain corresponding sample voltage phasor measurements that are used to calculate the threshold values. The sample measurements should not be spoofed.

The first column of the PMUBranches.csv consists of the bus indices that contain a PMU. The second column is each connecting and measured bus indices to this pmu bus. The third column is the corresponding current phasor TAG name of the RTAC C37.118 client device. The fourth column consists of the phasor TAG names of the RTAC C37.118 server devices which are used to output the corrected current phasors. The fifth column and beyond contain the corresponding sample current phasor measurements that are used to calculate the threshold values. The sample measurements should not be spoofed.

A sample simple network is given in figure 2. The corresponding PMUBuses.csv and PMUBranches.csv files are given in figure 3 and figure 4 respectively.

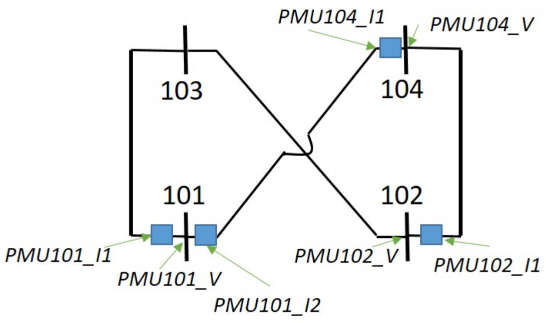
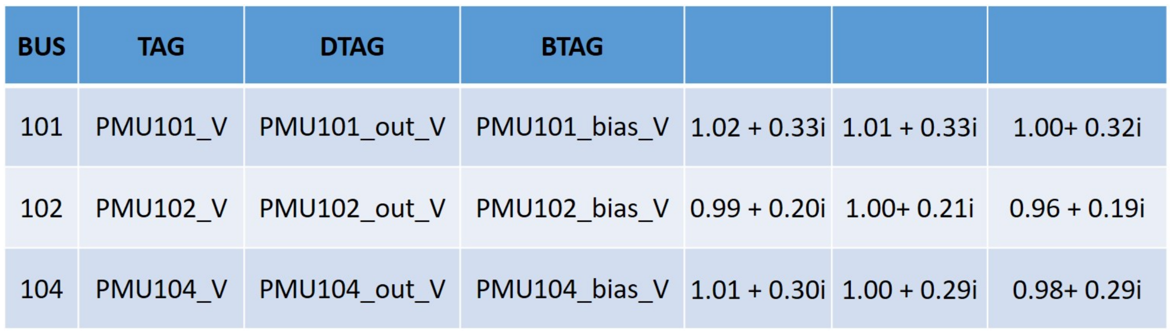


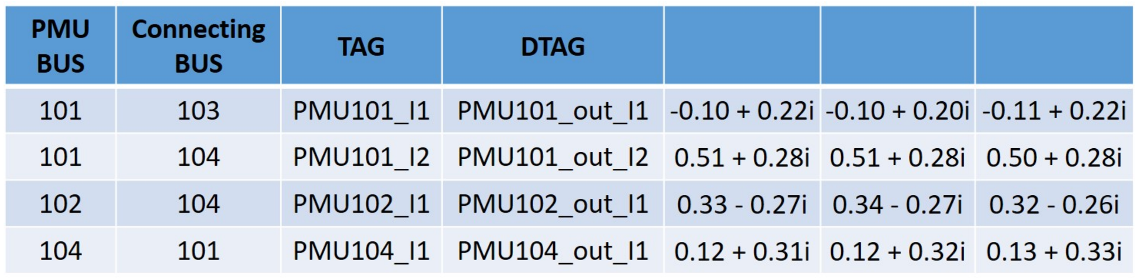
Figure 2: Sample PMU network

[1]. If the file names or column names are changed, then the corresponding names should also be changed in the python script



BUS	TAG	DTAG	BTAG			
101	PMU101_V	PMU101_out_V	PMU101_bias_V	1.02 + 0.33i	1.01 + 0.33i	1.00+ 0.32i
102	PMU102_V	PMU102_out_V	PMU102_bias_V	0.99 + 0.20i	1.00+ 0.21i	0.96 + 0.19i
104	PMU104_V	PMU104_out_V	PMU104_bias_V	1.01 + 0.30i	1.00 + 0.29i	0.98+ 0.29i

Figure 3: Sample PMUBuses.csv file



PMU BUS	Connecting BUS	TAG	DTAG			
101	103	PMU101_I1	PMU101_out_I1	-0.10 + 0.22i	-0.10 + 0.20i	-0.11 + 0.22i
101	104	PMU101_I2	PMU101_out_I2	0.51 + 0.28i	0.51 + 0.28i	0.50 + 0.28i
102	104	PMU102_I1	PMU102_out_I1	0.33 - 0.27i	0.34 - 0.27i	0.32 - 0.26i
104	101	PMU104_I1	PMU104_out_I1	0.12 + 0.31i	0.12 + 0.32i	0.13 + 0.33i

Figure 4: Sample PMUBranches.csv file

2.2 Python script

The given python script reads the input files as described in section 2.1 and processes them to generate necessary input files for RTAC as in section 2.3. The python script required pandas and numpy packages. The first step in the script is to change the bus numbers to internal indexing. Specifically, the script first changes the bus numbers for consecutive values starting at 1. The next step is to generate the system Y bus. We follow the same AC modeling as described in [3]. One of the main prepossessing steps is to decompose the network into several zones. The overall zone calculation steps are given in algorithm 1.

The algorithm starts with a bus index that has a PMU. Then, it adds all the buses that are reachable to the initial bus. Here bus *i* and bus *j* are reachable if there exists a path between them. A path is a set of distinct measured branches that join the sequence of distinct buses. In other words, there exist at least one branch between two different zones that are not measured. The algorithm stops when it covers all the buses. After identifying the zones, the algorithm extracts some useful information such as the number of PMUs, number of measurements, modal matrix, Index of voltage phasor in measurement vector, and threshold values for each zone. It also extracts the

Algorithm 1 Network Decomposition

```
Initialize : visitedBus $\leftarrow \emptyset$, zones $\leftarrow \emptyset$, numZone = 0, pmuBus $\leftarrow$ {all buses
with PMUs}
For all $i \in$ pmuBus do
    zoneBus $\leftarrow \emptyset$, neighbourBus $\leftarrow \emptyset$
    If $i \notin$ visitedBus
        numZone = numZone +1
        visitedBus $\leftarrow$ visitedBus $\cup \{i\}$
        zoneBus $\leftarrow$ zoneBus $\cup \{i\}$
        neighbourBus $\leftarrow$ {all measured branches neighbour pmus of $i$}
        For all $j \in$ neighbourBus do
            If $j \notin$ visitedBus
                visitedBus $\leftarrow$ visitedBus $\cup \{j\}$
                zoneBus $\leftarrow$ zoneBus $\cup \{j\}$
                neighbourBus $\leftarrow$ neighbourBus
                        $\cup$ {all measured branches neighbour pmus of $j$}
            neighbourBus $\leftarrow$ neighbourBus$\setminus \{j\}$
        zones $\leftarrow$ {zoneBus}
Output zones, numZone
```

submatrix of the matrix *H* for each zone.

It is important to notice the construction of the measurement vector of the zone. It is because the measurement vector and the matrix **H** should follow the measurement modal as in equation (9). We arrange the measurement into a vector as follows. Starting with the lowest bus index with a PMU in a zone the first measurement is its voltage phasor, then we add all the current phasors which this bus is the from bus according to the branch matrix. After that, we add all the current phasors which this bus is the to bus according to the branch matrix. We follow this arrangement for each bus with PMU and concatenate them together to create the measurement vector of the zone. Finally, we concatenate the measurement vector for each zone to construct the total measurement vector. We follow the same procedure to arrange the corrected measurement in the RTAC implementation. Thus we arrange the RTAC C37.118 client and server tags in the same way.

Another important calculation that implement in the script is threshold calculation for each zone. After identify the zones, the script extract the corresponding non spoofed sample measurements from the PMUBuses.csv and PMUBranches.csv files. We then calculate the squared norm of the residue values. Then we arrange the squared norm of the residue values in ascending order. We picked the value that is corresponding to the given detection probability. For example if the ordered squared norm of the residue values vector is $\tilde{\mathbf{r}}$ and the detection probability is 0.99 then the threshold value is given by $\tilde{\mathbf{r}}[0.99 \times \text{length}(\tilde{\mathbf{r}})]$.

2.3 Output files

The script saves the results as .csv files that can be used as input to RTAC library. The generated files are,

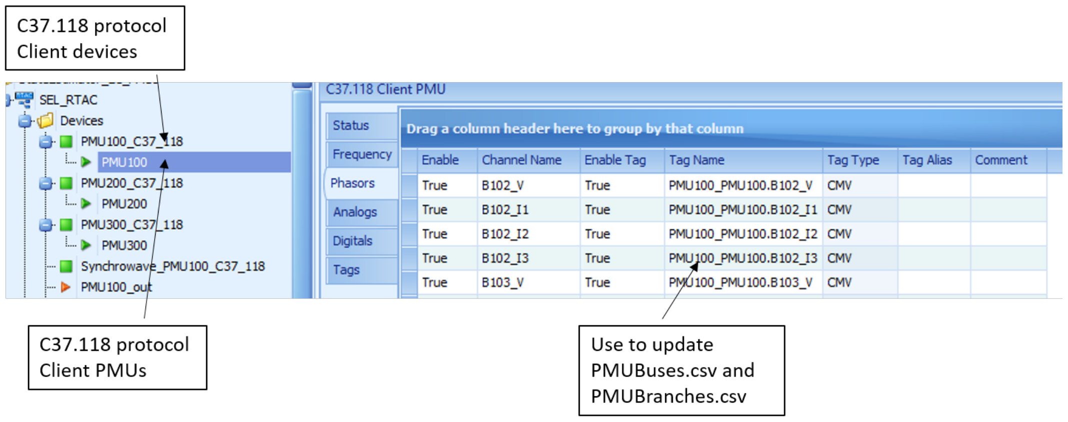
- **RTACCode** : RTAC code portion to use in main program
- **RTACTAGProcessor** : RTAC Tag processor destination tag names and Source expression
- **Nzones** : Number of identified zones in the network
- **Nmeasurements** : Number of measurements in reach zone
- **Npmus** : Number of PMUs in each zone
- **Thresholds** : Threshold values for each zone
- **Zonenames** : File names of PMU indices and OrthoH matrices
- **Zone_pmus_indexi** : Set of indices in zone i measurement vector
- **Zone_OrthoH_matrixi** : Model matrix of zone i

The RTACCode.csv file consists of a code portion that can be used to create a measurement vector. The first column consists of each entry of the measurement vector variable in RTAC. The second column contains the corresponding C37.118 client devices TAG names. This measurement vector should feed as an input to the data correction library. The RTACTAGProcessor.csv contains RTAC tag processor destination tag names and Source expressions in the first and second columns respectively. It is used to map the variable in RTAC that represents the presence of the attack, corrected phasor measurements, and estimated angle biases to the corresponding C37.118 server devices TAG names. As shown in figure 1, all other .csv files (Nzones.csv, Nmeasurements.csv, Npmus.csv, Thresholds.csv, Zonenames.csv, Zone_pmus_indexi.csv and Zone_OrthoH_matrixi.csv) should upload to the RTAC file manager.

3 RTAC Implementation

3.1 Setting up SEL RTAC

The user needs to input PMU measurements as a vector for a given time point to the library. To collect the PMU measurements at the RTAC, the user can configure C37.118 protocol client devices (Other -> C37.118 Protocol -> Connection Type:client-Ethernet). The tag names of the phasors of this client PMUs should be given to the python script as described in section 2. A sample C37.118 client devices is given in figure (5)


C37.118 protocol Client devices
SEL_RTAC
Devices
PMU100_C37_118
PMU100
PMU200_C37_118
PMU200
PMU300_C37_118
PMU300
Synchrowave_PMU100_C37_118
PMU100_out
C37.118 Client PMU
Status
Frequency
Phasors
Analogs
Digitals
Tags
Drag a column header here to group by that column
C37.118 protocol Client PMUs
Use to update PMUBuses.csv and PMUBranches.csv

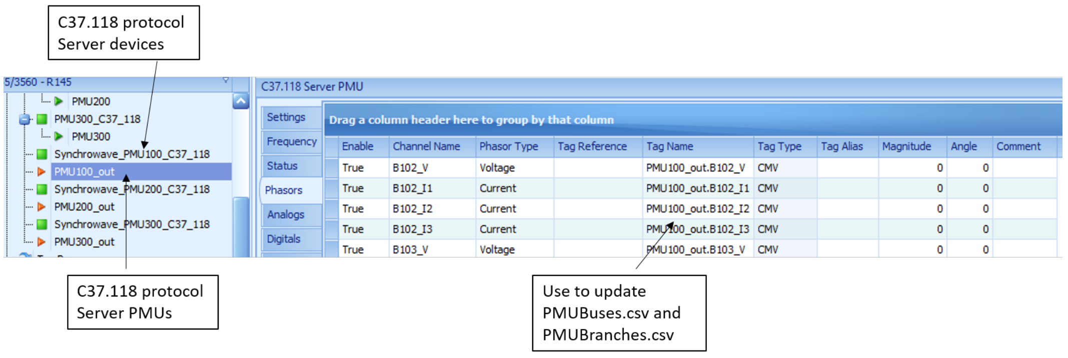
Enable	Channel Name	Enable Tag	Tag Name	Tag Type	Tag Alias	Comment
True	B102_V	True	PMU100_PMU100.B102_V	CMV		
True	B102_I1	True	PMU100_PMU100.B102_I1	CMV		
True	B102_I2	True	PMU100_PMU100.B102_I2	CMV		
True	B102_I3	True	PMU100_PMU100.B102_I3	CMV		
True	B103_V	True	PMU100_PMU100.B103_V	CMV		

Figure 5: A sample C37.118 client devices setup

The RTAC can send the corrected measurements as PMU phasors to any other outside device. To send the measurements, the user needs to configure C37.118 protocol server devices (Other -> C37.118 Protocol -> Connection Type:server-Ethernet). The tag names of the phasors of this server PMUs should be given to the python script as described in section 2. A sample C37.118 client devices is given in figure (6). For more details about setting up these devices, please refer AcSELerator RTAC Instruction Manual.

3.2 Data correction Algorithm

The user must provide the measurement vector (i.e., y_values) obtained by using the RTAC code portion from the python script. RTAC library uses all other required information from the .csv files uploaded to the RTAC file manager. We propose a greedy iterative algorithm to solve the optimization problem (12). The pseudo-code of the data correction algorithm for a particular zone γ is given in the algorithm (2).


C37.118 protocol Server devices
5/3560 - R145
PMU200
PMU300_C37_118
PMU300
Synchrowave_PMU100_C37_118
PMU100_out
Synchrowave_PMU200_C37_118
PMU200_out
Synchrowave_PMU300_C37_118
PMU300_out
C37.118 Server PMU
Settings
Frequency
Status
Phasors
Analogs
Digitals
Drag a column header here to group by that column
C37.118 protocol Server PMUs
Use to update PMUBuses.csv and PMUBranches.csv

Enable	Channel Name	Phasor Type	Tag Reference	Tag Name	Tag Type	Tag Alias	Magnitude	Angle	Comment
True	B102_V	Voltage		PMU100_out.B102_V	CMV		0	0	
True	B102_I1	Current		PMU100_out.B102_I1	CMV		0	0	
True	B102_I2	Current		PMU100_out.B102_I2	CMV		0	0	
True	B102_I3	Current		PMU100_out.B102_I3	CMV		0	0	
True	B103_V	Voltage		PMU100_out.B103_V	CMV		0	0	

Figure 6: A sample C37.118 server devices setup

The variable isAttack is a Boolean variable to indicate the presence of an attack. In the beginning, we assume there is no attack and set it to 0. We initialized the estimated angle bias $\hat{\alpha}^{(t),\gamma}$ to zero. The set $\mathcal{A}^{[itr],\gamma}$ denotes the support of the estimated angle bias vector. Support of a vector is the index set of non-zero elements of that vector. We start the algorithm with empty set $\mathcal{A}^{[0],\gamma}$. The first step is to identify one possible spoofed PMU location (i.e., index of non-zero location in the angle bias vector $\alpha^{(t),\gamma}$). To do that as in step 1 we calculate the normalized projection residue magnitudes $\tilde{r}_i^{\gamma}$ that are corresponding to each PMU i. Here $\mathbf{r}_i^{[itr],\gamma}$ denote the projection residue of measurements from PMU i. We considered only PMUs that have not been identified as spoofed in previous iterations. Then we pick the PMU that has the highest normalized projection residue magnitude in step 2. We update the support with this newly identified PMU location as in step 3. In step 4 we estimate the angle bias by constraining the support to the estimated support. Finally, in step 5 we calculate the residue vector with the estimated angle biases. We continue the algorithm until the squared L2 norm of the residue is less than the pre-define threshold value τ^{γ}. After the algorithm is complete, we can calculate the corrected measurements using estimated angle biases. The algorithm output the corrected PMU measurements, estimated angle bias vector, and the Boolean variable to indicate the presence of the attack.

3.3 PMUDataCorrector Library

The PMUDataCorrector Library perform spoofing attack detection and correction for incoming PMU phasor measurements.

Supported Firmware Versions

You can use this library on any device configured using ACSELERATOR

Algorithm 2 Data correction algorithm

```
Require: $\hat{\alpha}^{(t),\gamma} = 0, \mathbf{r}^{[0],\gamma} = (\mathbf{I}_{m^\gamma} - \mathbf{P}_{H^\gamma})\phi_\gamma^{-1}(\hat{\alpha}^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma}, \mathcal{A}^{[0],\gamma} = \emptyset, itr = 1$, isAttack = 0
  while $\|\mathbf{r}^{[itr-1],\gamma}\|_2^2 > \tau^\gamma$ do
  isAttack = 1
  1. Compute normalized residue magnitude
     $$\tilde{r}_i^\gamma = \frac{\|\mathbf{r}_i^{[itr-1],\gamma}\|_2^2}{m_i^\gamma} \quad \forall i \in \mathcal{T}^\gamma \backslash \mathcal{A}^{[itr-1],\gamma}$$
  2. Select the largest normalized residue magnitude
     $$i^* = \underset{i \in \mathcal{T}^\gamma \backslash \mathcal{A}^{[itr-1],\gamma}}{\text{argmax}} \ \tilde{r}_i^\gamma$$
  3. Update the support of $\alpha^{(t),\gamma}$
     $$\mathcal{A}^{[itr],\gamma} \leftarrow \mathcal{A}^{[itr-1],\gamma} \cup \{i^*\}$$
  4. Compute the estimate $\hat{\alpha}^{(t),\gamma}$
     $$\hat{\alpha}^{(t),\gamma} = \underset{\alpha^{(t),\gamma}: supp(\alpha^{(t),\gamma}) \subseteq \mathcal{A}^{[itr],\gamma}}{\text{argmin}} (\mathbf{I}_{m^\gamma} - \mathbf{P}_{H^\gamma})\phi_\gamma^{-1}(\alpha^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma}$$
  5. Update the residual
     $$\mathbf{r}^{[itr],\gamma} = (\mathbf{I}_{m^\gamma} - \mathbf{P}_{H^\gamma})\phi_\gamma^{-1}(\hat{\alpha}^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma}$$
  6. $itr = itr + 1$
  end while
  Corrected Data : $\hat{\mathbf{z}}^{(t),\gamma} = \phi_\gamma^{-1}(\hat{\alpha}^{(t),\gamma})\bar{\mathbf{z}}^{(t),\gamma}$
  Output $\hat{\mathbf{z}}^{(t),\gamma}$ and $\hat{\alpha}^{(t),\gamma}$, isAttack
```

RTAC® SEL-5033 Software with firmware version R145 or higher.

Global Constants

This section lists values of global constants provided for facilitating work with the library.

Name	IEC 61131 Type	Value	Description
Length_max_large_GB	UINT	5000	Large length
Length_max_medium_GB	UINT	2000	Medium length
Length_max_low_GB	UINT	100	Small length
max_Nzones	UINT	10	Maximum number of zones

Classes

This section enumerates the classes used to access the functionality of this library.

class_LoadData(Class)

This class reads all the Zone_pmus_index and Zone_OrthoH_matrix .csv files. It creates two base vectors. voltage phasor indices and modal matrix arrays for each zone.

Inputs

Name	IEC 61131 Type	Description
Npmus	ARRAY [0 .. Length_max_low_GB-1] OF INT	Number of PMUs for each zone
Nmeasurement	ARRAY [0 .. Length_max_low_GB-1] OF INT	Number of measurements for each zone
zone_names	ARRAY [0 .. Length_max_low_GB-1] OF STRING	CSV file names for each zone
Nzones	UINT	Number of zones

Outputs

fun_ReadFromFileIntM(Method)

IEC 61131 Type	Description
DynamicVectors.class_BaseVector (Length_max_medium_GB *SIZEOF(UINT),max_Nzones)	Vector with PMU indices arrays
DynamicVectors.class_BaseVector (Length_max_large_GB *SIZEOF(struct_ComplexRect) ,max_Nzones)	Vector with Modal matrices arrays

This method reads .csv file with medium number of integers.

Inputs

Name	IEC 61131 Type	Description
FileName	STRING	File name to read
Length	UINT	Length of the file to read

Outputs

IEC 61131 Type	Description
ARRAY[0..Length_max_medium_GB-1] OF INT	Contend of the file

fun_ReadFromFileComplexL(Method)

This method reads csv file with large number of complex numbers.

Inputs

Name	IEC 61131 Type	Description
FileName	STRING	File name to read
Length	UINT	Length of the file to read

Outputs

IEC 61131 Type	Description
ARRAY[0..Length_max_large_GB-1] OF struct_ComplexRect	Contend of the file

Functions

fun_DataCorrection(function)

This function detects and corrects the input PMU measurement of a zone. Then It will return the corrected PMU measurements, Estimated angle biases, and presence of the attack for the particular zone.

Inputs

Name	**IEC 61131 Type**	**Description**
N_PMUs	UINT	Number of PMUs
N_measurements	UINT	Number of measurements
threshold	struct_ComplexRect	Threshold value
pmu_indices	ARRAY[0 .. Length_max_medium_GB-1] OF UINT	Voltage phasor index
orthoH_values	ARRAY [0 .. Length_max_large_GB-1] OF struct_ComplexRect	Modal matrix
y_values	ARRAY [0 .. Length_max_large_GB-1] OF struct_ComplexRect	Measurement vector

Outputs

IEC 61131 Type	**Description**
ARRAY[0..Length_max_large_GB-1] OF vector_t	Corrected measurements
BOOL	Attack indicator
ARRAY [0 .. Length_max_low_GB-1] OF REAL	Estimated angle bias

fun_MainDataCorrection(function)

This function goes through each zone and performs the data correction. Then it will return the corrected PMU measurements, Estimated angle biases, and presence of the attack on the entire network.

Inputs

Outputs

Name	**IEC 61131 Type**	**Description**
Nzones	UINT	Number of Zones
Npmus	ARRAY[0..Length_max_low_GB-1] OF INT	Number of PMUs for each zone
Nmeasurement	ARRAY[0..Length_max_low_GB-1] OF INT	Number of measurements for each zone
threshold_mat	ARRAY[0..Length_max_low_GB-1] OF struct_ComplexRect	Threshold value for each zone
baseVector_ PMU_indices	DynamicVectors.class_BaseVector (Length_max_medium_GB *SIZEOF(UINT),max_Nzones)	Vector with PMU indices arrays
baseVector_ orthoH_values	DynamicVectors.class_BaseVector (Length_max_large_GB *SIZEOF(struct_ComplexRect) ,max_Nzones)	Vector with Modal matrices arrays

IEC 61131 Type	**Description**
ARRAY [0 .. Length_max_large_GB-1] OF CMV	Corrected measurements for entire system
BOOL	Attack indicator for entire system
ARRAY [0 .. Length_max_low_GB-1] OF REAL	Estimated angle bias for entire system

fun_Find(function)

This function returns the number of non zero entries in an array.

Inputs

Name	IEC 61131 Type	Description
y	ARRAY [0 .. Length_max_low_GB-1] OF BYTE	nonzero values

Outputs

IEC 61131 Type	Description
UINT	Number of non zero values

fun_SetDiff(function)

This function returns the set of entries in the array [1 .. idx] with the given length idx which are not in the input array y.

Inputs

Name	IEC 61131 Type	Description
y	ARRAY [0 .. Length_max_low_GB-1] OF BYTE	Input array to compare
idx	UINT	Given length

Outputs

IEC 61131 Type	Description
ARRAY [0 .. Length_max_low_GB-1] OF BYTE	set of entries not in input

fun_ReadFromFileComplexS(function)

This method reads .csv file with small number of complex numbers.

Inputs

Outputs

Name	IEC 61131 Type	Description
FileName	STRING	File name to read
Length	UINT	Length of the file to read

IEC 61131 Type	Description
ARRAY[0..Length_max_low_GB-1] OF struct_ComplexRect	Contend of the file

fun_ReadFromFileIntS(function)

This method reads .csv file with small number of integers.

Inputs

Name	IEC 61131 Type	Description
FileName	STRING	File name to read
Length	UINT	Length of the file to read

Outputs

IEC 61131 Type	Description
ARRAY[0..Length_max_low_GB-1] OF UINT	Contend of the file

fun_ReadFromFileStringS(function)

This method reads .csv file with small number of strings.

Inputs

Name	IEC 61131 Type	Description
FileName	STRING	File name to read
Length	UINT	Length of the file to read

Outputs

Examples

These examples demonstrate the capabilities of this library. Do not mistake them as suggestions or recommendations from SEL.

IEC 61131 Type	Description
ARRAY[0..Length_max_low_GB-1] OF Strings	Contend of the file

Implement the best practices of your organization when using these libraries. As the user of this library, you are responsible for ensuring correct implementation and verifying that the project using these libraries performs as expected.

Spoofing attack detection and correction

Objective

The objective of this example is to detect and correct the spoofing attacked PMU measurements. The corrected PMU measurements, a signal to indicate the presence of an attack and estimated angle biases are sent to the outside device.

Assumptions

This example assumes the following are true:

- The RTAC project incorporates PMUDataCorrector library.
- The RTAC is collecting synchrophasor measurements from several C37.118 client devices. The user has prepared the input .csv files accordingly and uploaded the relevant output .csv files to the RTAC file manager as described in the section 2.
- The RTAC is sending synchrophasor measurements from several C37.118 server devices. The user has prepared the input .csv files accordingly and updated the TAG processor using RTACTAGProcessor.csv files as described in the section 2.
- The synchrophasor message rate is 60 samples per second.
- The RTAC Main Task cycle time, under which all programs are being run, is set to 16 ms.

Solution

The Library returns a Boolean number to indicate an attack (0: no attack and 1: attack), corrected measurements, and the estimated biases for all the

Code Snippet 1: Global variables

```
VAR_GLOBAL
  isAttack : BOOL;
  y_hat_send : ARRAY [0 .. Length_max_large_GB-1] OF CMV;
  solution_deg_final : ARRAY [0 .. Length_max_low_GB-1] OF REAL;
END_VAR
```

PMUs. To collect these values the user can configure the following global variables as given in the code snippet 1.

- isAttack : Indicate an attack
- y_hat_send : Corrected measurements
- solution_deg_final : Estimated biases

The python script first decomposes the network into several zones. Then it generates several useful information and saves them as .csv files. Before performing the data correction the user needs to read these .csv files into RTAC. To read these .csv files the user needs to define several variables as given in the code snippet 2. In code snippet 2, "y_values" is used as an array to collect incoming PMU measurements. If the user changes this variable name then they should also update the python script with the same variable name in the RTACCode.csv file generating code portion. The variable "Data_inst" is an instant of class "class_LoadData". The variable "Firstscan" is a boolean variable to identify the first run of the RTAC code execution.

After defining the variables the user can read the .csv files in the RTAC file manager and send them to the RTAC data correction function as given in code snippet 3. Since the content of the .csv files does not change with time, the .csv files are read into RTAC only during the first cycle. Then the user can copy and paste the content of the RTACCode.csv file to generate the input PMU measurement array. The final portion of the program is to call the data correction function with all the necessary input values.

Code Snippet 2: Data Correction variables

```
PROGRAM prg_data_correction

VAR
Data_inst : PMUDataCorrector.class_LoadData;

y_values : ARRAY [0 .. Length_max_large_GB-1] OF struct_ComplexRect;

Firstscan :BOOL:= TRUE;
temp :ARRAY[0..Length_max_low_GB-1] OF INT;

Nzones : UINT;
Npmus : ARRAY[0..Length_max_low_GB-1] OF INT;
Nmeasurement : ARRAY[0..Length_max_low_GB-1] OF INT;
zone_names : ARRAY[0..Length_max_low_GB-1] OF STRING;
threshold_mat : ARRAY[0..Length_max_low_GB-1] OF struct_ComplexRect;

END_VAR
```

4 Test-bed setup

Testing the Data Correction Library (PMUDataCorrector) can be performed on a test-bed with the minimum required components shown in Figure 7. This section will detail one implementation to test the system. The basic equipment required is a SEL RTAC, a device capable of simulating a power system and outputting synchrophasor data, and a device to receive data from the RTAC. The PMUDataCorrector was developed and tested on a SEL-3555 RTAC. The SEL-3505 RTAC does not have the necessary computational power to run the PMUDataCorrector.

Software and hardware versions used for the testbed:

- RT-Lab version 2021.2.0.244
- RTDS OS version Red Hat 5.2 (2.6.29.6-opalrt-6.2.1)
- RTAC firmware R145
- AcSELerator RTAC software version 1.33.149.12000
- Synchrowave Operations 1.3.0.33
- Synchrowave Operations running on Ubuntu 18.04.6 LTS

All files described here can be found on our github repository.

Code Snippet 3: Data Correction program

```
IF Firstscan THEN

(*// read number of zones*)
temp :=  PMUDataCorrector.fun_ReadFromFileIntS('Nzones.csv', 1);
Nzones := INT_TO_UINT(temp[0]);

(*//read total vector of number of pmus in each zone*)
Npmus :=  PMUDataCorrector.fun_ReadFromFileIntS('Npmus.csv', Nzones);

(*//read total vector of number of measurement in each zone*)
Nmeasurement :=  PMUDataCorrector.fun_ReadFromFileIntS('Nmeasurements
    .csv', Nzones);

(*//read total vector of threshold values in each zone*)
threshold_mat :=  PMUDataCorrector.fun_ReadFromFileComplexS('
    Thresholds.csv', Nzones);

(*//read file names. One zone has two file names*)
zone_names :=  PMUDataCorrector.fun_ReadFromFileStringS('Zonenames.
    csv', 2*Nzones);

Data_inst(Npmus := Npmus,Nmeasurement:=Nmeasurement,zone_names:=
    zone_names,Nzones:=Nzones);

Firstscan := FALSE;

ELSE

(*//Insert the RTAC Code portion here*)

y_hat_send := PMUDataCorrector.fun_MainDataCorrection(Nzones,Npmus,
    Nmeasurement,threshold_mat,Data_inst.baseVector_PMU_indices,
    Data_inst.baseVector_orthoH_values,y_values,temp_isAttack=>
    isAttack,temp_solution_deg_final=>solution_deg_final);

END_IF
```

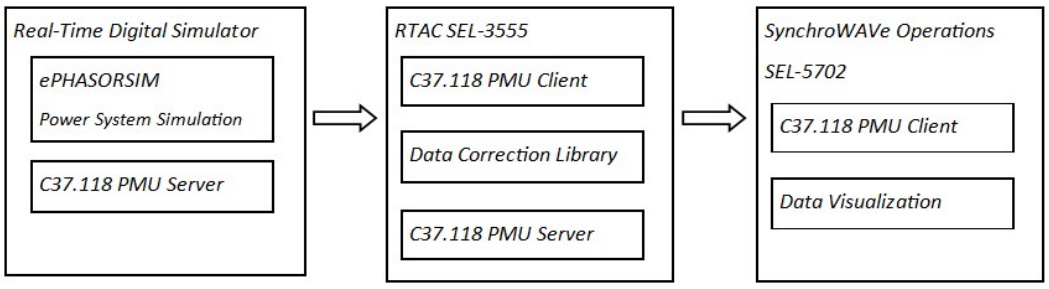


Figure 7: A complete testbed to test PMUDataCorrector

4.1 Power System Simulation

A power system simulation is used to generate synchrophasor data. The implementation described here uses an Opal-RT OP5600 Real-Time Digital Simulator (RTDS), with Siemens PTI Power System Simulation (PSSE) input files.

The RTDS is configured through Opal's software, RT-Lab, but uses Matlab Simulink to create and edit models to be simulated. One option for power system simulation is to use the ePHASORSIM block for Matlab Simulink as shown in Figure 8. To use this, the power system network and associated dynamic machine data can be input in the form of Siemens PTI PSSE files, Eaton CYME files, or directly input into an excel file. This example uses PSSE v32 files. Regardless of input type, the "pins" excel file must be created to define the inputs and outputs from the ePHASORSIM block. Figure 9 shows the inputs for the ePHASORSIM block. This is only an overview of the required components and basic setup. See the relevant software for full explanation of setup and configuration.

It is necessary to ensure that all the input files to ePHASORSIM are stable and working in their respective programs. In this demonstration, PSSE input files are used and were tested for stability with no contingencies for several minutes. Another important note in working with ePHASORSIM is that modifying the time-step of the simulation or the time-step of the ePHASORSIM block can lead to incorrect outputs. It was noted in testing that this resulted in the bus voltage magnitude being around 2.0 pu.

The "pins" file is an excel document necessary to define inputs and outputs to ePHASORSIM. Figure 10 shows the first few columns of the document. Column C and beyond extract specific simulation quantities for use in the Simulink model. See the RT-Lab documentation for further details.

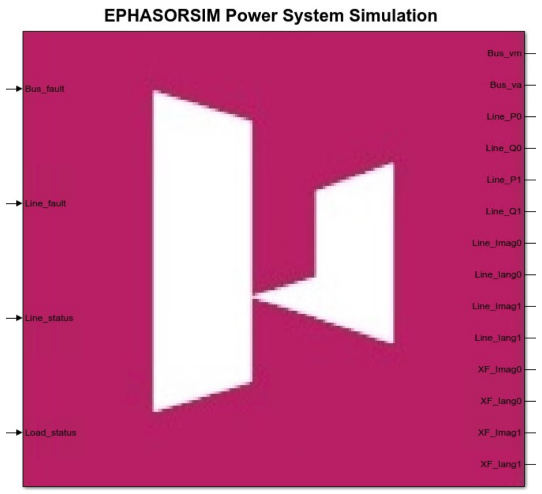
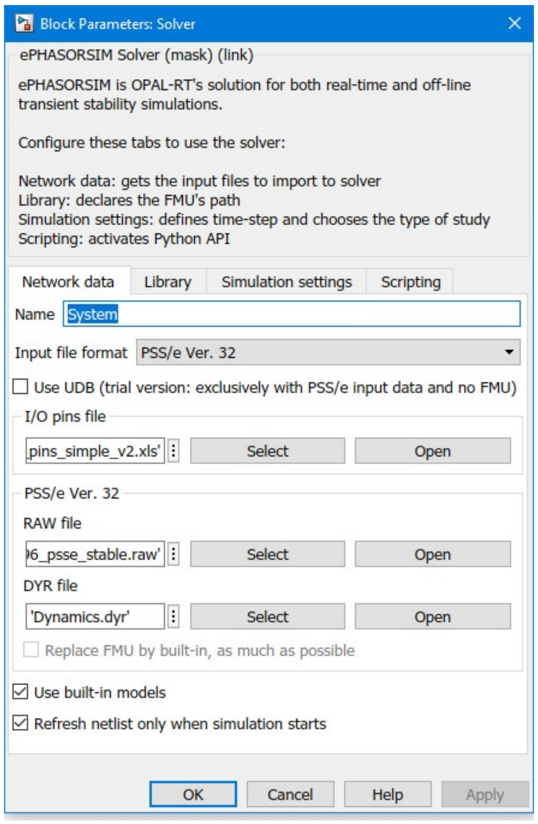


Figure 8: Opal-RT ePHASORSIM block in Matlab Simulink with inputs and outputs configured


Block Parameters: Solver

ePHASORSIM Solver (mask) (link)

ePHASORSIM is OPAL-RT's solution for both real-time and off-line transient stability simulations.

Configure these tabs to use the solver:

Network data: gets the input files to import to solver
Library: declares the FMU's path
Simulation settings: defines time-step and chooses the type of study
Scripting: activates Python API

Network data | Library | Simulation settings | Scripting

Name: System

Input file format: PSS/e Ver. 32

☐ Use UDB (trial version: exclusively with PSS/e input data and no FMU)

I/O pins file: pins_simple_v2.xls' | Select | Open

PSS/e Ver. 32

RAW file:)6_psse_stable.raw' | Select | Open

DYR file: 'Dynamics.dyr' | Select | Open

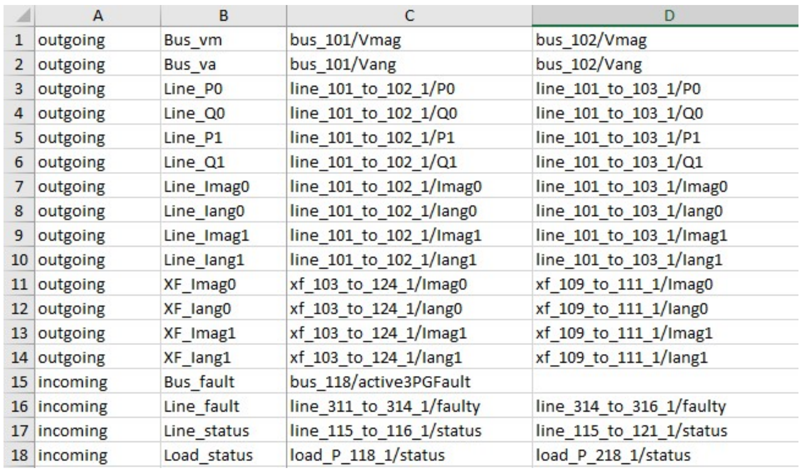
☐ Replace FMU by built-in, as much as possible

☑ Use built-in models

☑ Refresh netlist only when simulation starts

OK | Cancel | Help | Apply

Figure 9: Inputs to Opal-RT ePHASORSIM block in Matlab Simulink



	A	B	C	D
1	outgoing	Bus_vm	bus_101/Vmag	bus_102/Vmag
2	outgoing	Bus_va	bus_101/Vang	bus_102/Vang
3	outgoing	Line_P0	line_101_to_102_1/P0	line_101_to_103_1/P0
4	outgoing	Line_Q0	line_101_to_102_1/Q0	line_101_to_103_1/Q0
5	outgoing	Line_P1	line_101_to_102_1/P1	line_101_to_103_1/P1
6	outgoing	Line_Q1	line_101_to_102_1/Q1	line_101_to_103_1/Q1
7	outgoing	Line_Imag0	line_101_to_102_1/Imag0	line_101_to_103_1/Imag0
8	outgoing	Line_Iang0	line_101_to_102_1/Iang0	line_101_to_103_1/Iang0
9	outgoing	Line_Imag1	line_101_to_102_1/Imag1	line_101_to_103_1/Imag1
10	outgoing	Line_Iang1	line_101_to_102_1/Iang1	line_101_to_103_1/Iang1
11	outgoing	XF_Imag0	xf_103_to_124_1/Imag0	xf_109_to_111_1/Imag0
12	outgoing	XF_Iang0	xf_103_to_124_1/Iang0	xf_109_to_111_1/Iang0
13	outgoing	XF_Imag1	xf_103_to_124_1/Imag1	xf_109_to_111_1/Imag1
14	outgoing	XF_Iang1	xf_103_to_124_1/Iang1	xf_109_to_111_1/Iang1
15	incoming	Bus_fault	bus_118/active3PGFault	
16	incoming	Line_fault	line_311_to_314_1/faulty	line_314_to_316_1/faulty
17	incoming	Line_status	line_115_to_116_1/status	line_115_to_121_1/status
18	incoming	Load_status	load_P_118_1/status	load_P_218_1/status

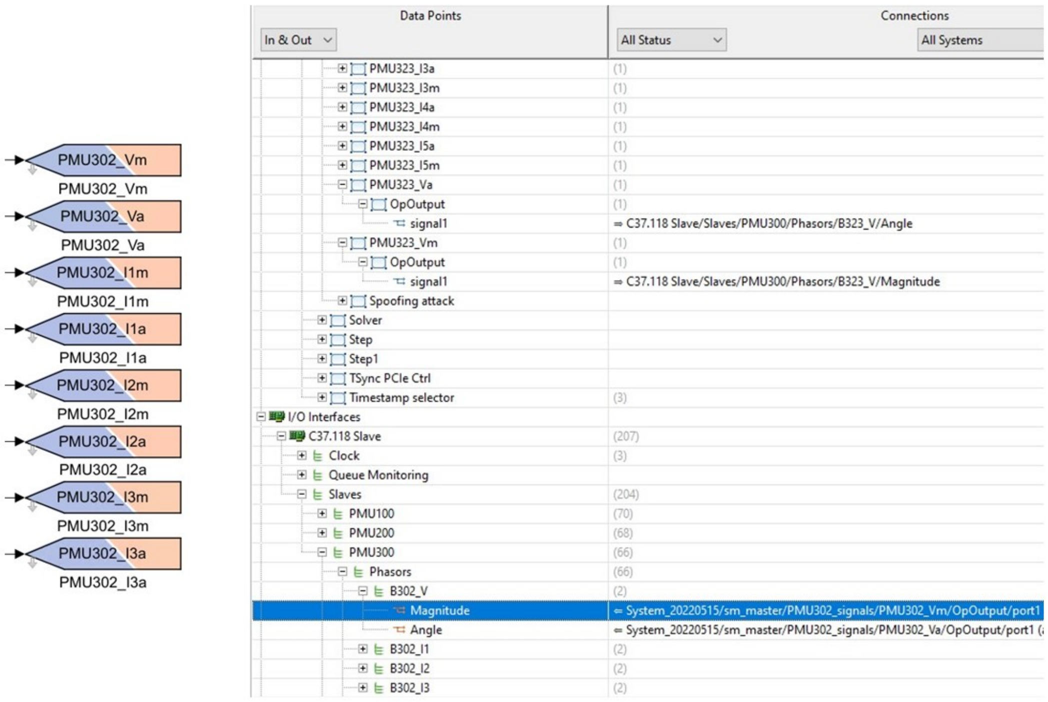
Figure 10: "Pins" Excel file that defines ePHASORSIM inputs and outputs

4.2 Requirements of Synthetic Data

The data must be considered valid by the data visualization platform, Synchrowave Operations in this case. Data that contains invalid flags may not be available from within the Synchrowave platform. The data quality flag must be "good" and "clockFailure" must be false or the data will not show up. Thus, these two requirements must be considered when designing the system to send data to the RTAC. Some modifications may be made by the RTAC to correct some data deficiencies, but timestamps are particularly difficult to modify and correct. Significant testing on Oregon State University's testbed revealed few ways to achieve all the necessary requirements.

There are alternatives to the PMU data generation already presented, but these produce data that do not meet the aforementioned requirements. For example, there is RT-Lab block that simplifies the creation of a PMU server device. However, all SEL PMU client devices tested (RTAC, Synchrowave Operations, PDC) considered the data as "invalid" which limited usage in downstream devices.

The testbed described here uses a Spectracom Tsync-PCIe card built into the RTDS and receives IRIG timecodes over a coaxial connection from a SEL-2488 GPS Network Clock. There is a RT-Lab block in Matlab Simulink for accessing the data from the clock. This is required to add valid timestamps to the PMU data. It is highly recommended to check the data being sent from the data generation platform and ensure it is considered valid and has a good timestamp before attempting any further configuration.



```
Data Points                              Connections
In & Out                                 All Status          All Systems
      PMU323_I3a                         (1)
      PMU323_I3m                         (1)
      PMU323_I4a                         (1)
      PMU323_I4m                         (1)
      PMU323_I5a                         (1)
      PMU323_I5m                         (1)
      PMU323_Va                          (1)
        OpOutput                         (1)
          signal1                        ⇒ C37.118 Slave/Slaves/PMU300/Phasors/B323_V/Angle
      PMU323_Vm                          (1)
        OpOutput                         (1)
          signal1                        ⇒ C37.118 Slave/Slaves/PMU300/Phasors/B323_V/Magnitude
      Spoofing attack
    Solver
    Step
    Step1
    TSync PCIe Ctrl
    Timestamp selector                   (3)
I/O Interfaces
  C37.118 Slave                          (207)
    Clock                                (3)
    Queue Monitoring
    Slaves                               (204)
      PMU100                             (70)
      PMU200                             (68)
      PMU300                             (66)
        Phasors                          (66)
          B302_V                         (2)
            Magnitude                    ⇐ System_20220515/sm_master/PMU302_signals/PMU302_Vm/OpOutput/port1
            Angle                        ⇐ System_20220515/sm_master/PMU302_signals/PMU302_Va/OpOutput/port1 (
          B302_I1                        (2)
          B302_I2                        (2)
          B302_I3                        (2)
```

Figure 11: Left: Matlab Simulink output blocks. Right: RT-Lab configuration to connect Simulink outputs to C37.118 driver

4.3 C37.118 Synchrophasor Communication

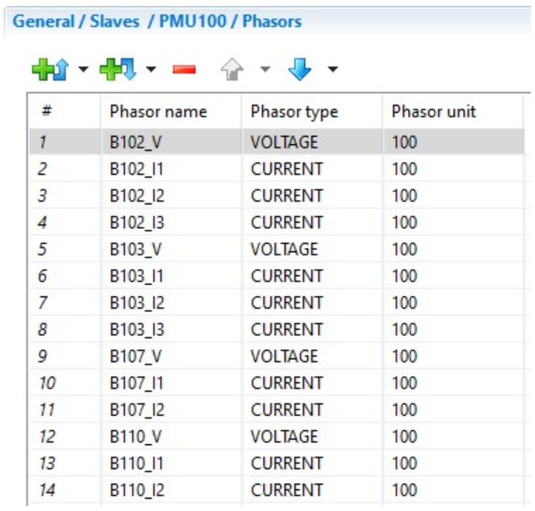
To send synchrophasor data, the data points from Simulink must be connected to relevant data points in the C37.118 driver in RT-Lab through the project configuration. This is achieved by creating PMU devices in the RT-Lab I/O Interfaces and adding the required PMU slaves or servers. The PMU connection details, PMU name and phasor names defined here need to match exactly the configuration of the RTAC, or data will not be transmitted.

Figure 11, shows the Matlab outputs connected to the RTDS C37.118 driver. The names for the slaves shown here are defined in the C37.118 configuration window, see Figure 12.

4.4 RTAC Configuration

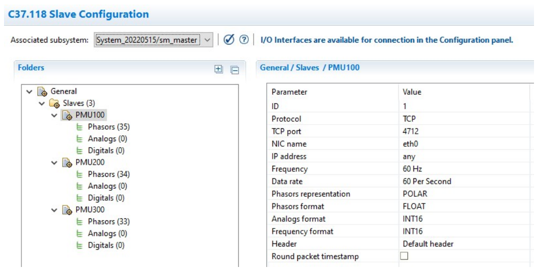
The SEL-3555 RTAC is used to receive the synchrophasor data over C37.118 protocol, perform the data correction functions, then output the synchrophasor data to the visualization system. The SEL RTAC is configured through the AcSELerator RTAC software and through a web browser.

The connections and names configured in the RTDS must match exactly


General / Slaves / PMU100 / Phasors

#	Phasor name	Phasor type	Phasor unit
1	B102_V	VOLTAGE	100
2	B102_I1	CURRENT	100
3	B102_I2	CURRENT	100
4	B102_I3	CURRENT	100
5	B103_V	VOLTAGE	100
6	B103_I1	CURRENT	100
7	B103_I2	CURRENT	100
8	B103_I3	CURRENT	100
9	B107_V	VOLTAGE	100
10	B107_I1	CURRENT	100
11	B107_I2	CURRENT	100
12	B110_V	VOLTAGE	100
13	B110_I1	CURRENT	100
14	B110_I2	CURRENT	100

Figure 12: Sample of RTDS C37.118 Phasor tags


C37.118 Slave Configuration

Associated subsystem: System_20220515/sm_master

I/O Interfaces are available for connection in the Configuration panel.

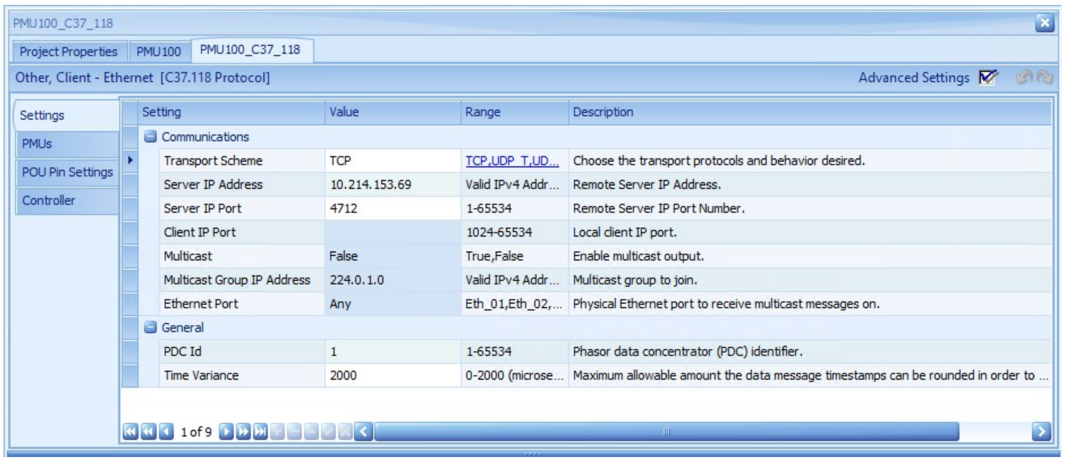
Folders

- General
  - Slaves (3)
    - PMU100
      - Phasors (35)
      - Analogs (0)
      - Digitals (0)
    - PMU200
      - Phasors (34)
      - Analogs (0)
      - Digitals (0)
    - PMU300
      - Phasors (33)
      - Analogs (0)
      - Digitals (0)

General / Slaves / PMU100

Parameter	Value
ID	1
Protocol	TCP
TCP port	4712
NIC name	eth0
IP address	any
Frequency	60 Hz
Data rate	60 Per Second
Phasors representation	POLAR
Phasors format	FLOAT
Analogs format	INT16
Frequency format	INT16
Header	Default header
Round packet timestamp	☐

Figure 13: RTDS C37.118 Configuration


PMU100_C37_118

Project Properties | PMU100 | PMU100_C37_118

Other, Client - Ethernet [C37.118 Protocol]

Advanced Settings

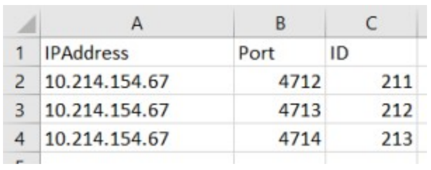
Setting	Value	Range	Description
Communications			
Transport Scheme	TCP	TCP,UDP_T,UD...	Choose the transport protocols and behavior desired.
Server IP Address	10.214.153.69	Valid IPv4 Addr...	Remote Server IP Address.
Server IP Port	4712	1-65534	Remote Server IP Port Number.
Client IP Port		1024-65534	Local client IP port.
Multicast	False	True,False	Enable multicast output.
Multicast Group IP Address	224.0.1.0	Valid IPv4 Addr...	Multicast group to join.
Ethernet Port	Any	Eth_01,Eth_02,...	Physical Ethernet port to receive multicast messages on.
General			
PDC Id	1	1-65534	Phasor data concentrator (PDC) identifier.
Time Variance	2000	0-2000 (microse...	Maximum allowable amount the data message timestamps can be rounded in order to ...

Figure 14: RTAC PMU Client configuration

what is configured in the RTAC. To receive the data, C37.118 client devices must be created for each PMU configured in the RTDS. Be sure to choose the appropriate type of communication (serial or Ethernet). Figure 14 shows the configuration window for an RTAC PMU client. Note that the "Time Variance" field is set to 2000 microseconds, this is due to how the data is timestamped at the RTDS. If this field is set to 0 microseconds, most of the data will be missed by the RTAC. There is an option on the RTDS to "Round packet timestamp," although this does not fully resolve the issue of missed data. As such, this option was left unchecked in RT-Lab in favor of using only the "Time Variance" setting in the RTAC.

Outputs from the RTAC are configured similarly to the inputs. The outputs are configured as C37.118 server devices. A tag list, that contains phasors, analogs and digitals, must be manually created for each device and associated to each connection. For this example, the output phasor names were named exactly the same as the input phasor names. The angle measurement received by each PMU is forwarded without modification to Synchrowave Operations by creating an additional phasor for each PMU and appending "_raw" to each. In addition to the input phasor data, analogs were also added to send the bias data. Bias data is the difference between the received angle measurement and the corrected angle measurement. A digital (or boolean) is also configured to send the spoofing attack status to the visualization system.

Inputs from PMU devices are extracted directly into global variables for usage in the Data Correction Library (PMUDataCorrector). Outputs from the PMU devices in the RTAC are configured through the Tag Processor in the RTAC. It is recommended to set the "Live Data Enabled" as viewable for at least a few data points in the tag processor. This allows for a fast check



	A	B	C
1	IPAddress	Port	ID
2	10.214.154.67	4712	211
3	10.214.154.67	4713	212
4	10.214.154.67	4714	213

Figure 15: Synchrowave Connections file

that the data is being received by the RTAC by viewing it on the RTAC web interface.

One issue that may be encountered is the deadband in the RTAC. This example used data in pu and as such the RTAC ignored the data completely because the deadband was set to 100. The deadband settings are hidden by default in the tag processor. Right click on the column headings in the tag processor and choose the deadband fields to show them. Ensure they are set appropriately for the system.

4.5 PMU Data Visualization

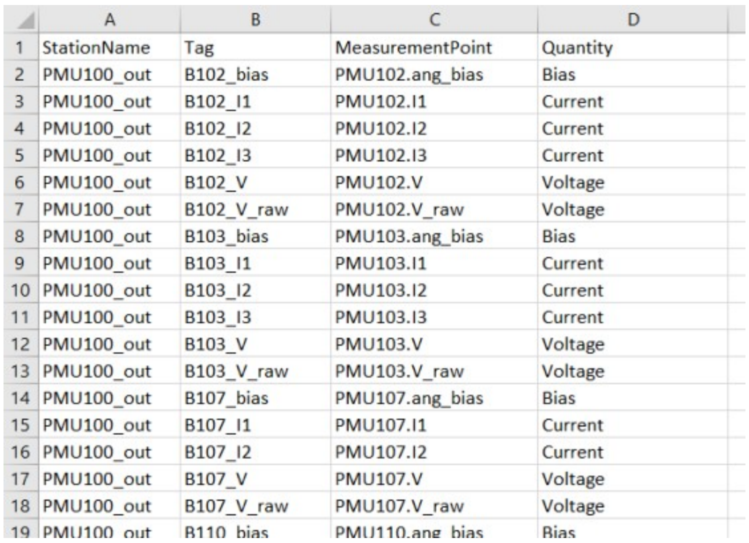
The data is visualized using SEL-5702 Synchrowave Operations. To make a complete visualization the C37.118 reader, signal monitor, and one-line diagram must be configured. The configurations below assume that the user has some familiarity with the Synchrowave Operations. Google Chrome was used to interface with Synchrowave Operations.

4.5.1 C37.118 Reader

Inputs to Synchrowave are configured by providing the IP address, port, and ID number for each server PMU in the "connection.csv" configuration file, Figure 15. After allowing time for C37.118 reader application to restart the C37.118 configuration needs to be exported. Upon exporting the configuration, Synchrowave will automatically gather the tags being transmitted and save them in the "SignalMappings.csv" file as shown in Figure 16. The SignalMappings file then needs to finished by completing the MeasurementPoint and Quantity fields. These fields are what is published to the Synchrowave system and are the names that will be used in graphical objects discussed in following sections.

4.5.2 Signal Monitor

The signal monitor is used to raise a general alarm when a spoofing attack has been detected by the RTAC. In this example, it was configured as a digital



	A	B	C	D
1	StationName	Tag	MeasurementPoint	Quantity
2	PMU100_out	B102_bias	PMU102.ang_bias	Bias
3	PMU100_out	B102_I1	PMU102.I1	Current
4	PMU100_out	B102_I2	PMU102.I2	Current
5	PMU100_out	B102_I3	PMU102.I3	Current
6	PMU100_out	B102_V	PMU102.V	Voltage
7	PMU100_out	B102_V_raw	PMU102.V_raw	Voltage
8	PMU100_out	B103_bias	PMU103.ang_bias	Bias
9	PMU100_out	B103_I1	PMU103.I1	Current
10	PMU100_out	B103_I2	PMU103.I2	Current
11	PMU100_out	B103_I3	PMU103.I3	Current
12	PMU100_out	B103_V	PMU103.V	Voltage
13	PMU100_out	B103_V_raw	PMU103.V_raw	Voltage
14	PMU100_out	B107_bias	PMU107.ang_bias	Bias
15	PMU100_out	B107_I1	PMU107.I1	Current
16	PMU100_out	B107_I2	PMU107.I2	Current
17	PMU100_out	B107_V	PMU107.V	Voltage
18	PMU100_out	B107_V_raw	PMU107.V_raw	Voltage
19	PMU100_out	B110_bias	PMU110.ang_bias	Bias

Figure 16: Synchrowave SignalMappings file defining incoming tags and associated names in Synchrowave

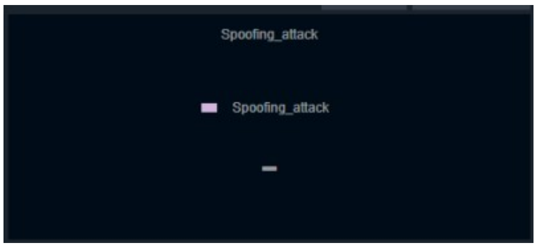
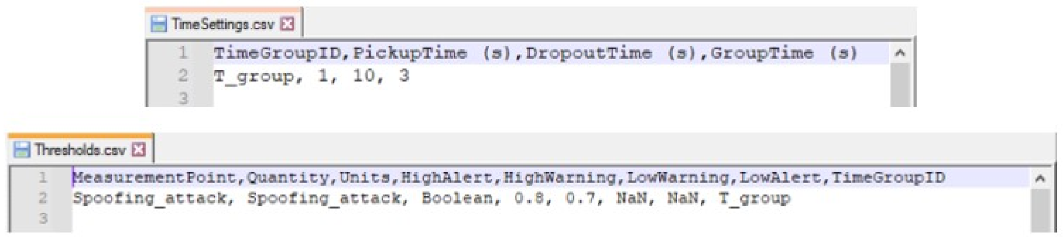


Figure 17: A Synchrowave display panel for indicating if a spoofing attack is occurring

and sent on the C37.118 connection. The MeasurementPoint and Quantity in the "SignalMappings.csv" file were defined as **Spoofing_attack**. To get Spoofing_attack to indicate when an attack is occurring, create a numeric dashboard panel and choose Spoofing_attack as the measurement point. It will look like Figure 17.

To configure the Signal Monitor, export the Signal Monitor configuration and edit as shown in Figure 18. The "TimeSettings.csv" file defines how Synchrowave handles the alerts. The "Thresholds.csv" file defines how the alert is triggered. The low warning and low alert are not configured because the normal state for the spoofing attack variable is 0 and the attack state is 1. Reimport the signal monitor configuration and now the panel in Figure 17 will flash red when a spoofing attack has been detected and publish an alert to the notifications system.



TimeSettings.csv

```
TimeGroupID,PickupTime (s),DropoutTime (s),GroupTime (s)
T_group, 1, 10, 3
```

Thresholds.csv

```
MeasurementPoint,Quantity,Units,HighAlert,HighWarning,LowWarning,LowAlert,TimeGroupID
Spoofing_attack, Spoofing_attack, Boolean, 0.8, 0.7, NaN, NaN, T_group
```

Figure 18: Synchrowave Signal Monitor configuration for triggering alerts from spoofing attacks

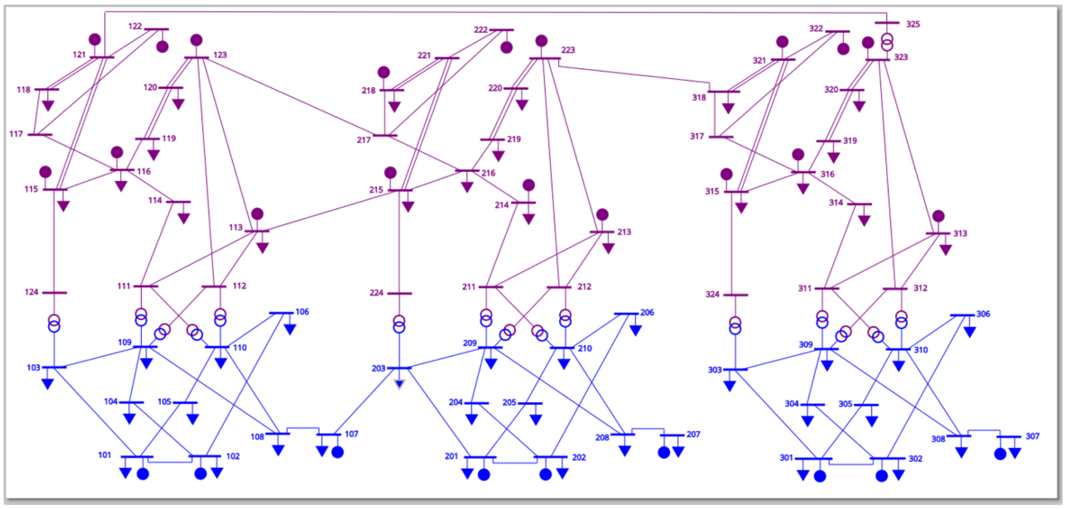


Figure 19: IEEE RTS1996 Test Case one line diagram

4.5.3 One-Line Diagram

The one-line diagram panel offers more functionality than a basic one-line. Synchrowave only accepts one-line diagrams as SVG (Scalable Vector Graphics) files. For this example, a one-line was created for the IEEE RTS96 test case, using the geographically distributed example from the case documentation. Any SVG editor can be used.

To provide better situational awareness to the user, the one-line diagram was modified so that when a spoofing attack is detected at a particular bus a red bar shows up over the bus to indicate the issue as shown in Figure 20.

This is achieved by creating a rectangle over each bus that has spoof detection, then adding additional parameters to the SVG object, Figure 21 shows the required modifications. The basic steps are to add an additional attribute titled **data-signal-driven-attributes**, then add the code shown in the red box on Figure 21. Note that the signal name must match exactly the values from the C37.118 configuration shown in Figure 16. The signal name

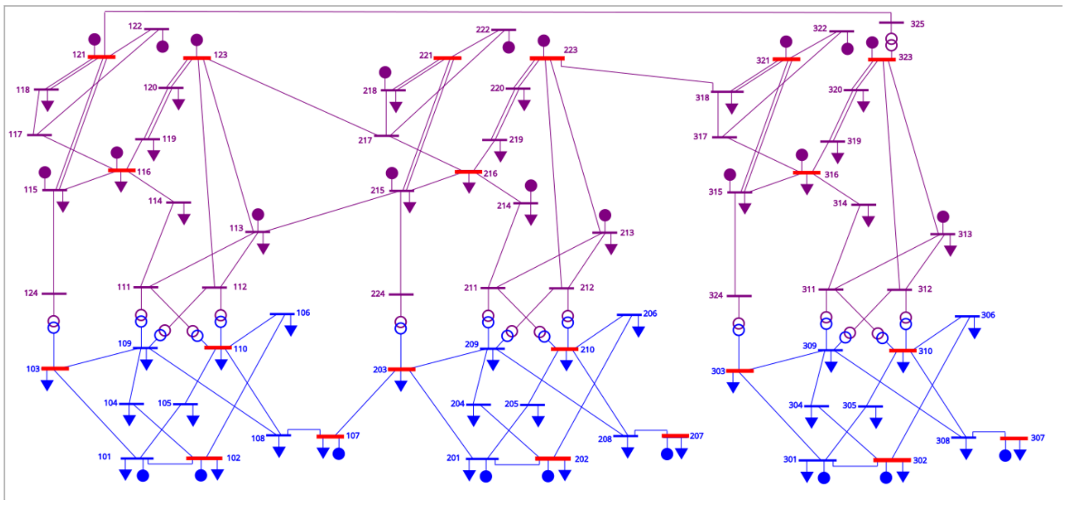


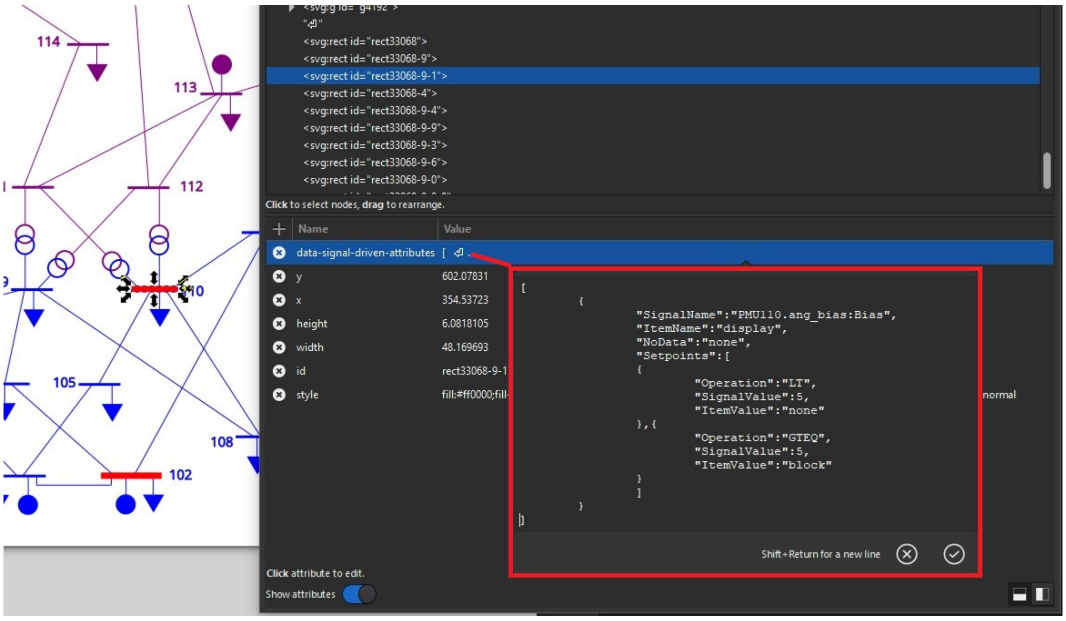
Figure 20: IEEE RTS1996 Test Case one line diagram with indicators for detected spoofing attacks at monitored buses

is made up of "MeasurementPoint:Quantity". "Operation" indicates the logical test that is being performed. "LT" is less than, and "GTEQ" is greater than or equal to. The SignalValue is set to 5, which means that if there is greater or equal to 5 degrees of difference between the measured angle and the corrected angle then the red bar is shown over the bus. This process was repeated for the negative values of bias so the indicators show for positive and negative spoofing attacks.

5 Examples

We performed a full test bed validation using RTS 1996 test power system. The network information is saved in PSSE readable .raw file format. We use psse2mpc function in MATPOWER [2] to convert .raw file to MATPOWER case struct. The BusMatrix.csv and BranchMatrix.csv are created by copying and pasting the mpc.bus and mpc.branch matrices respectively. The test system consists of 73 buses and 120 branches. Out of 73 buses, we placed PMUs at 21 buses. The buses with PMUs are 102, 103, 107, 110, 116, 121, 123, 202, 203, 207, 210, 216, 221, 223, 302, 303, 307, 310, 316, 321 and 323. We assumed that all the branches associated with these PMU buses are measured. Thus there are 81 measured branches and the PMU network provides 102 phasor measurements (21 voltage phasors for each bus and 81 current phasors for each branch).

We prepared the input files as described in subsection 2.1 using PSSE readable .raw file format. The python script preprocesses the input and



```
<svg:g id="g4192">
"⏎"
<svg:rect id="rect33068">
<svg:rect id="rect33068-9">
<svg:rect id="rect33068-9-1">
<svg:rect id="rect33068-4">
<svg:rect id="rect33068-9-4">
<svg:rect id="rect33068-9-9">
<svg:rect id="rect33068-9-3">
<svg:rect id="rect33068-9-6">
<svg:rect id="rect33068-9-0">
```

Click to select nodes, drag to rearrange.

Name	Value
data-signal-driven-attributes	[⏎ .
y	602.07831
x	354.53723
height	6.0818105
width	48.169693
id	rect33068-9-1
style	fill:#ff0000;fill- … normal

```
[
        {
                "SignalName":"PMU110.ang_bias:Bias",
                "ItemName":"display",
                "NoData":"none",
                "Setpoints":[
                {
                        "Operation":"LT",
                        "SignalValue":5,
                        "ItemValue":"none"
                },{
                        "Operation":"GTEQ",
                        "SignalValue":5,
                        "ItemValue":"block"
                }
                ]
        }
]
```

Figure 21: Modifying one line diagram so that the red indicators are only visible when bus is being spoofed

generates necessary files for the RTAC library. We used python 3.7.4 with pandas and numpy packages for this experiment. The zone decomposition algorithm decomposes the network into two zones. The first zone consist of 14 PMU buses including buses 102, 103, 107, 110, 123, 202, 203, 207, 210, 216, 221, 223, 316 and 321. There are 68 phasor measurements associated with this zone. The second zone consists of 7 PMU buses including buses 116, 121, 302, 303, 307, 310, and 323. There are 34 phasor measurements associated with this zone. All the input and output files for this experiment can be found in the github repository.

We tested the system under three different operating conditions. The first normal operation shows the system when no spoofing attack is occurring. The data correction system is still active. Next, a spoofing attack is demonstrated. The system detects the spoofing attack, corrects the data, and alerts the operator. Lastly, the system is demonstrated during a spoofing attack but with no data correction or detection. The first step is to configure the RTDS to generate synchrophasors and send the data to the RTAC. We followed the procedure described in subsection 4.1, 4.2, and 4.3. Unless specified otherwise, all tests use the following parameters:

- RTDS time-step is 0.001 seconds
- GPS spoofing limited to +/- 20 degrees

- GPS spoofing ramp is set to 1 degree/second
- PMUs 116,102,221 and 323 are under spoofing attack
- All spoofing attacks affect the substations uniformly
- All spoofing attacks begin at the same time
- PMU communication set to 60 samples/second
- No faults occur during the simulations
- Additive Gaussian noise to magnitude signals 5e-4
- Additive Gaussian noise to angle signals 0.06

We follow the procedure in subsections 3.1 and 4.4 to setup three C37.118 client devices to receive synchrophasors from RTDS and three C37.118 server devices to send the corrected synchrophasors to Synchrowave Operations. A part of the setup is shown in figure 5 and 6. The configuration of client and server devices are shown in figures 13 and 14.

In order to send the RTAC library results, we defined three global variables as given in code snippet 1. After loading the PMUDataCorrector library, we followed the procedure in code snippets 2 and 3 to create the main program. We set the cycle time to 16ms and watchdog time to 45000ms in the Main controller. We used the RTACTAGProcessor.csv file to update the RTAC Tag processor.

We used SEL-5702 Synchrowave Operations to visualize the results. We followed the same procedure as in subsection 4.5 to setup the Synchrowave Operations.

5.1 Normal Operation

Figure 22 shows the complete visualization when no spoofing attack is present. Note how the phase angles are closely grouped in the system, no indicators are present at any buses in the one-line diagram, the biases remain 0 or very close, and that no alarms are present.

5.2 A Spoofing Attack with Data Correction

Figure 23 shows the complete visualization dashboard is shown during a spoofing attack under the parameters listed at the beginning of this section. Notice that the spoofing attack indicator is showing up red triggering an alarm.

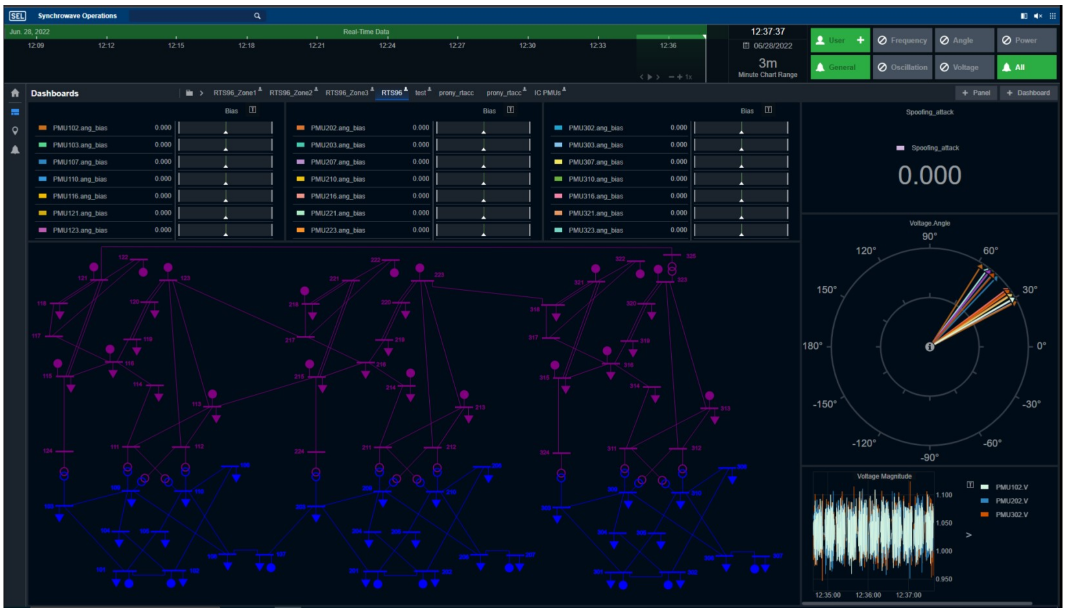


Figure 22: Complete Synchrowave visualization during normal operation

The bias bar graphs are showing where the spoofing attack is occurring. This is reiterated by the indicators on the one-line diagram.

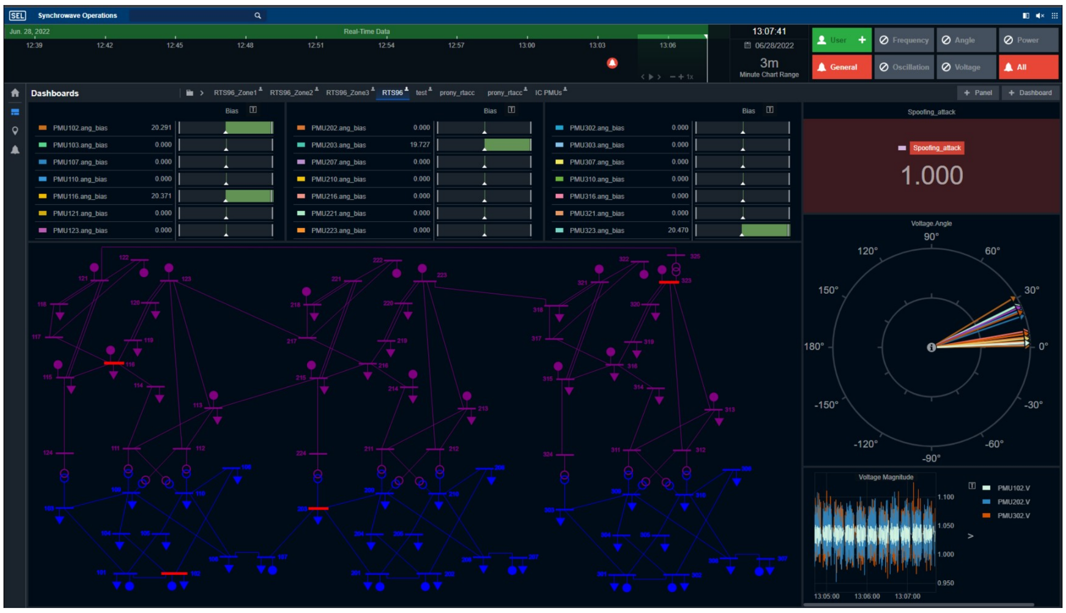
The visualization in Figure 23 clearly shows buses 102, 116, 203, and 323 are being spoofed in the positive direction. The parameters in the RTDS were configured for the same buses. Take note of the phase angle graph. All bus phase angles have remained tightly grouped demonstrating the data correction is keeping the phase angles in the normal state despite the presence of an ongoing spoofing attack.

The visualization in Figure 24 shows the data correction correcting a positive attack on buses 102 and 116, and a negative attack on buses 221 and 323.

5.3 A Spoofing Attack with No Data Correction

Figure 25 demonstrates what occurs during a spoofing attack when the data correction library is not used. Note that a few voltage angles have moved away from the rest of the group in the phase angle graph. Compare this to Figure 23, which is experiencing the same attack on buses 102, 116, 203 and 323 in which the phase angles have been corrected and have not separated from the group.

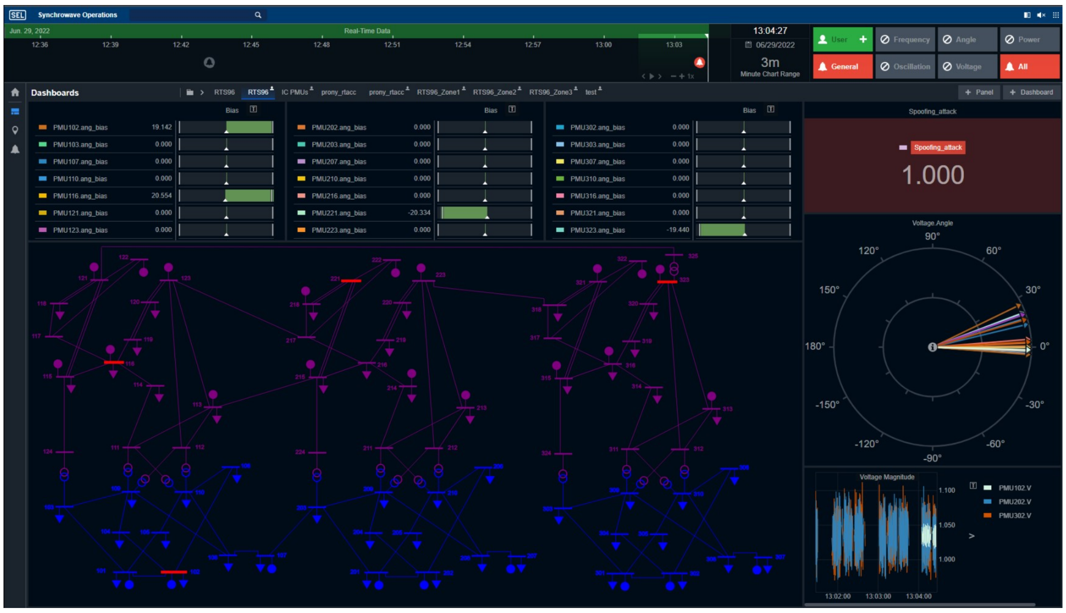
No alerts are generated, the bias remains 0 and the one-line diagram does not show any indications because the PMUDataCorrector is offline. Note that the limits for the GPS spoofing attack were changed to +/- 180 degrees so



Bias		Bias		Bias	
PMU102.ang_bias	20.291	PMU202.ang_bias	0.000	PMU302.ang_bias	0.000
PMU103.ang_bias	0.000	PMU203.ang_bias	19.727	PMU303.ang_bias	0.000
PMU107.ang_bias	0.000	PMU207.ang_bias	0.000	PMU307.ang_bias	0.000
PMU110.ang_bias	0.000	PMU210.ang_bias	0.000	PMU310.ang_bias	0.000
PMU116.ang_bias	20.371	PMU216.ang_bias	0.000	PMU316.ang_bias	0.000
PMU121.ang_bias	0.000	PMU221.ang_bias	0.000	PMU321.ang_bias	0.000
PMU123.ang_bias	0.000	PMU223.ang_bias	0.000	PMU323.ang_bias	20.470

Spoofing_attack 1.000

Figure 23: Complete Synchrowave visualization during a spoofing attack



Bias		Bias		Bias	
PMU102.ang_bias	19.142	PMU202.ang_bias	0.000	PMU302.ang_bias	0.000
PMU103.ang_bias	0.000	PMU203.ang_bias	0.000	PMU303.ang_bias	0.000
PMU107.ang_bias	0.000	PMU207.ang_bias	0.000	PMU307.ang_bias	0.000
PMU110.ang_bias	0.000	PMU210.ang_bias	0.000	PMU310.ang_bias	0.000
PMU116.ang_bias	20.554	PMU216.ang_bias	0.000	PMU316.ang_bias	0.000
PMU121.ang_bias	0.000	PMU221.ang_bias	-20.334	PMU321.ang_bias	0.000
PMU123.ang_bias	0.000	PMU223.ang_bias	0.000	PMU323.ang_bias	-19.440

Spoofing_attack 1.000

Figure 24: Complete Synchrowave visualization during an attack with positive and negative biases

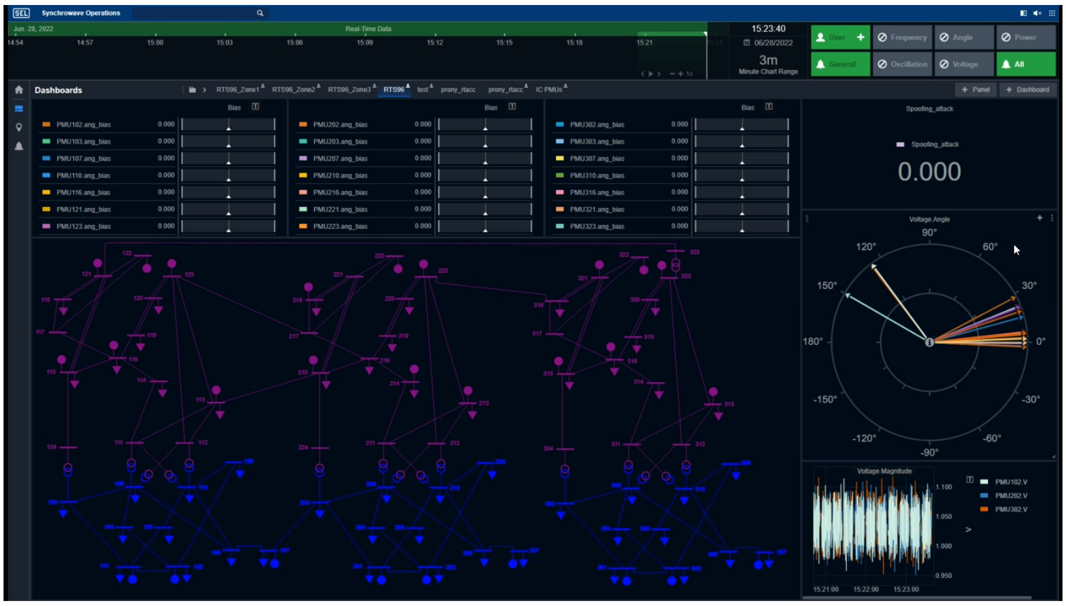


Figure 25: Complete Synchrowave visualization during a spoofing attack with no data correction or spoofing detection

that the separation is very clear.

References

[1] Shashini De Silva, Jinsub Kim, Eduardo Cotilla-Sanchez, and Travis Hagan. On pmu data integrity under gps spoofing attacks: A sparse error correction framework. *IEEE Transactions on Power Systems*, 36(6):5317–5332, 2021.

[2] Ray D. Zimmerman and Carlos E. Murillo-Sánchez. Matpower, December 2016. doi:10.5281/zenodo.3237810.

[3] Ray D. Zimmerman and Carlos E. Murillo-Sánchez. Matpower user's manual, December 2016. doi:10.5281/zenodo.3236526.