

System and Method of Detecting Attacks Against Automatic Generation Control (AGC) of a Grid

Technical Field

[0001] This disclosure generally relates to a method for detecting attacks against automatic generation control (AGC) of a grid. The present invention particularly relates to, but may not be limited to, identifying attacks using a pre-trained machine learning model applied to topological features of a point cloud representing data obtained from the AGC.

Background

[0002] Information and communication technologies (ICTs) are transforming critical systems, e.g., power grids, into cyber-physical systems. However, ICTs are vulnerable to cyberattacks.

[0003] Cyberattacks target the digital communications and control of a cyber-physical system (CPS) to cause the system damage. In power grids, false data injection (FDI) attack is one of the most dangerous cyberattacks against the system, where even a malware-infected PLC can send falsified sensor measurements that bypass standard detection schemes, i.e., bad data detection (BDD) [5], thus compromising the grid's state estimation which cause severe physical and economic damage.

[0004] Within the power system, cyberattacks against the grid's automatic generation control (AGC) [6–10] are especially dangerous. AGC adapts the output of generators to dynamically adjust system conditions in real time, to regulate the grid's frequency within a safe range. Breach of this range constitutes a *frequency excursion* that may damage generators if they are not quickly disconnected from the grid; both disconnection and damage can cause widespread and prolonged blackouts.

[0005] Addressing the problem of detecting FDI attacks in a robust and timely manner is particularly challenging in light of the stealthiness of such attacks when introduced by the advanced attacker. Detection of FDI attacks as well as other cyberattacks against mission-critical systems is an active area of research, and many solutions have been proposed for the CPS domain. These solution generally fall into two classes: model-based or data-driven approaches. The former requires a well-understood system model that

captures all the necessary details and yet is sufficiently tractable for fast solutions. Such models are usually unavailable for real-world systems, particularly those with significant interacting components and detailed parameters, like a power grid. In a data-driven solution, a system's normal vs. compromised behaviour manifests in the values of real-time sensor data streams from the system's operations. In this approach, machine learning (ML) over sensor data traces is used to build a model that determines whether an attack is underway.

[0006] A well recognized challenge for ML-based attack detection from streaming sensor data is how to identify the correlation of data in the temporal dimension, because subtle trends or relationships between data at successive time points are key, in addition to the data values themselves. Deep learning (DL) approaches to solving this problem allow information learned in a preceding time step to be selectively remembered and fed to a subsequent time step for dealing with cyberattacks. Such approaches usually have high complexity and take a long time to identify the attack.,

[0007] In view of the foregoing, it would be desirable to provide a system and method that remedies one or more of the abovementioned problems with prior art attack detection systems, or at least provides a useful alternative.

Summary

[0008] In light of the abovementioned problems, embodiments of the present invention focus on false data injection (FDI) attacks against automatic generation control, a critical system that maintains a power grid at its requisite frequency. The proposed methods detect the attack in power grids using topological data analysis (TDA) to extract key features from sensor data. These features serve as the input to a simple machine learning (ML) based detector for the attack.

[0009] In one embodiment, the present disclosure provides a method of detecting attacks against automatic generation control (AGC) of a grid. The method comprises obtaining data from the AGC. The data is pre-processed to generate a point cloud from which topological data analysis (TDA) features are extracted. Those TDA features are inputted to a machine learning (ML) model to generate a plurality of outputs. The outputs are interpreted to determine whether the data includes an attack.

[0010] In another embodiment, the present disclosure provides a system of detecting attacks against automatic generation control (AGC) of a grid, comprising a plurality of processors for:

- obtaining data from the AGC;
- pre-processing the data to generate a point cloud;
- extracting topological data analysis (TDA) features from the point cloud;
- inputting the TDA features to a machine learning (ML) model to generate a plurality of outputs; and
- interpreting the outputs to determine whether the data includes an attack

[0011] Advantageously, the invention employs a TDA-ML approach incorporating topological data analysis and ML to detect attacks robustly, in real-time. Compared to commonly used DL-based detection methods such as LSTM and other time-sequence feature extraction methods in ML, the present TDA-ML approach fundamentally improves attack detection and requires a much shorter data trace to reach a given level of effectiveness.

Brief Description of the Drawings

[0012] Some embodiments of systems and methods for detecting attacks against AGC of a grid, in accordance with present disclosure, will now be described, by way of non-limiting example only, with reference to the accompanying drawings in which:

[0013] Figure 1 illustrates a method for detecting attacks against AGC of a grid, in accordance with present teachings;

[0014] Figure 2 summarizes a three-phase TDA pipeline for time-sequence data;

[0015] Figure 3 illustrates a three-area 37-bus power grid;

[0016] Figure 4 schematically illustrates an overview of AGC;

[0017] Figure 5 is a diagram of the TDA-ML framework;

[0018] Figure 6 illustrates transforming a persistence diagram into a persistence landscape; and

[0019] Figure 7 shows a persistence diagram and corresponding first landscape in the time-optimal FDI attack in a 37-bus system.

Detailed Description

[0020] Embodiments described below employ a data-driven approach to attack detection. However, rather than relying on deep learning (DL) analysis in the temporal data dimension, present methods use topological data analysis (TDA) to obtain high-level data aggregates (i.e., beyond the low-level individual data points) as features to encode temporal characteristics of the data. These features are then used as inputs to a machine learning (ML) algorithm that does not need to be specifically designed for time-sequence data, such as Random Forest.

[0021] Motivated by the idea that data has shape, and the shape matters, the proposed TDA-based detector achieves better performance, especially in terms of the attack detection delay, than other time-sequence data feature extraction methods in ML, as well as the state-of-the-art deep learning LSTM detector. In particular, the TDA-based detector disclosed herein may detect newly-launched FDI attacks at least two times faster than other time-sequence data feature extraction methods in ML and the LSTM detector, enabling faster response and mitigation during real-time operation.

[0022] Embodiments of the TDA use concepts from algebraic topology to identify structure in data by capturing inherent qualitative information. TDA can provide an effective framework for data analysis and topological feature extraction, especially for high dimensional, noisy, or relatively sparse datasets, compared with other state-of-the-art time-sequence feature extraction methods where features are only related to statistical, temporal or spectral domains of the sequence.

[0023] Figure 1 illustrates a method 100 for detecting attacks against AGC of a grid. The method 100 broadly includes (step 102) obtaining data from the AGC. Data may be obtained either directly from the AGC control system (i.e. the system controlling AGC) or from a server in communication with the AGC control system or storing data acquired or received from the AGC control system. At step 104 the data is pre-processed to generate a point cloud. Topological data analysis (TDA) features are extracted from the point cloud at step 106. Step 108 involves inputting the TDA features to a machine learning (ML) model to generate a plurality of outputs. At step 110 the outputs are interpreted to determine whether or not the data includes an attack or attacks.

[0024] The present method 100 is designed to identify attacks in general. Although the discussion will be given with reference to FDI attacks, the present method 100 can

be used to identify other types of cyberattacks in power grids and other cyber physical systems. FDI attacks are considered in this disclosure since they are modern forms of attack, and introduce stealthiness to hide footprints of the attack, i.e., bypass the BDD and hide the footprint of the attack trace. The method 100 uses TDA-ML to capture anomalies and to detect these cyberattacks against AGC.

[0025] Figure 2 illustrates a processing pipeline for time-sequence analysis: forming a point cloud, creating a persistence diagram, and building a persistence landscape, in accordance with some embodiments. Step 102 involves obtaining data from sensors of an AGC control system performing or facilitating AGC. The data may be time series or time-sequence signals (200). To apply TDA to time-sequence data, the data is transformed to a point cloud per step 104. The point cloud is a finite set of points in a metric space (202). This transformation or conversion is a form of pre-processing of the data to facilitate its use in topological feature identification.

[0026] Many time-sequence signals are recorded in AGC. For illustrative purposes, for TDA-ML, the data obtained from the AGC at step 102 comprises area control error (ACE). ACE is used because of its correlation with frequency and tie-line measurements. AGC ensures that the power grid frequency lies in a narrow band around a set-point value. Each power grid is divided into separate areas, each usually operated by a different utility company, and AGC also regulates the power interchange among the different areas. Figure 3 illustrates a three-area power grid system 300 with 37 buses. The dotted lines 302 connecting two areas are called tie-lines. This 37-bus system is used for illustrative purposes as it is typical of a small- to mid-scale real-world grid. For this purpose, the AGC control system is a discrete-time AGC control system.

[0027] In a FDI attack, if an adversary knows the grid state measurement matrix $\mathbf{F}$, they can add an attack vector $\mathbf{a}$ to the power flow sensor measurements $\mathbf{z}$ in a manner that bypasses BDD with $\mathbf{a} = \mathbf{F}\mathbf{c}$, where $\mathbf{c}$ is an arbitrary vector. The power grid state will then be incorrectly estimated as $\hat{\mathbf{x}} + \mathbf{c}$. The power flow sensor measurement can be compromised by either attacking the sensor the sensor data communication links. For example, the adversary can compromise the grid's virtual private network (VPN) and then tamper with the data from multiple sensors.

[0028] In a time-optimal attack, the attacker's objective is to find an attack sequence that minimizes the time from the onset of an attack sequence to the first moment at which the average frequency deviation of all the areas (e.g. areas 1, 2 and 3 in Figure 3) reaches

the safety limits of the grid. In a stealthy FDI attack the attacker crafts actions to probe the system actively and learn its key parameters. Once sufficient information has been learned, the attacker selects and injects false data based on control theory to move the system's frequency precipitously to an unsafe region, leaving little time for the defender to react.

[0029] FDI attacks are launched at the tie-line in the areas i , i.e., p_E , by adding the attack vector $\mathbf{a}$ to cause the system frequency deviation. In summary, by compromising sensor readings (i.e., FDI attacks) in the AGC system, the adversary's objective is to cause the power system's frequency to breach the grid's frequency safety thresholds such that the damage will happen in the system. The safe frequency deviation range may be set as desired. The method 100 may assume a safe frequency deviation range of $[-0.5, 0.5]$ Hz, where any $\Delta\omega(t)$ out of this range as unsafe.

[0030] When obtaining data from AGC per step 102, which may contain injected false data intended to cause the frequency to breach a safety threshold, time is divided into slots – i.e. it is windowed. For AGC, ACE is an important feedback signal for regulating generators. For area i , the signal ACE_i is a weighted sum of frequency deviation ($\Delta\omega_i$) and power export deviation (Δp_{Ei}), i.e., $ACE_i = w_i^1 \Delta p_{Ei} + w_i^2 \Delta\omega_i$ with constant weights w_i^1 and w_i^2 . The control center sends ACE_i via the communication network to control the generator output in area i . The time required for one execution of the above control process varies from two to four seconds, and is referred to as one AGC cycle.

[0031] The sensors that measure the power export deviation Δp_{Ei} can be faulty and noisy, and state estimation (SE) is a common method to detect these kinds of data. Let $\mathbf{z} = [z_1, z_2, \dots, z_K]^T$ denote all the sensor measurements for power flow, where $[\cdot]^T$ denotes the vector/matrix transpose. Let $\mathbf{x} = [x_1, x_2, \dots, x_K]^T$ denote all states in power grids. The states are related by the sensor measurements with the equation $\mathbf{z} = \mathbf{F}\mathbf{x} + \mathbf{e}$, where $\mathbf{F}$ is a measurement matrix and $\mathbf{e}$ denotes the noise. SE estimates the states $\mathbf{x}$ by $\hat{\mathbf{x}} = (\mathbf{F}^T \mathbf{V} \mathbf{F})^{-1} \mathbf{V} \mathbf{z}$, where $\mathbf{V}$ is a weighted matrix. The estimated power export deviation can be obtained from $\hat{\mathbf{z}} = \mathbf{F} \hat{\mathbf{x}}$. An alarm will be raised if $\|\mathbf{z} - \hat{\mathbf{z}}\|$ is greater than a pre-defined threshold for bad data detection (BDD).

[0032] Figure 4 gives an overview of AGC 400. The control center 406 receives the sensor measurements $\mathbf{z}$ via the communication network 402 from the physical power system 404. The power export deviation Δp_{Ei} is obtained from state estimation (SE). With

the frequency deviation $\Delta\omega_i$ (received from the frequency sensors by the control center 406 over the communication network 402) and the computed power export deviation Δp_{Ei} , ACE_i is calculated and sent to the corresponding generator in physical power system 404 via the communication network 402.

[0033] As mentioned above and with reference to process flow 500 in Figure 5, data generated in accordance with AGC described in Figure 4, is obtained at step 502. The data is then pre-processed at 504. During data pre-processing the data is split or sampled using a sliding window of size W (506). The sliding window splits the data trace (i.e. data from AGC) into blocks of size W . For the training phase, training data is used in which the attack samples (sliding window blocks) are given a positive label and non-attack samples (i.e. normal samples) are given a negative label.

[0034] The data is then converted to a point cloud per 508. Assuming there are N measurements of ACE for a N -area power grid system: let x_t^i denote the time-series signal ACE_i for the i^{th} area. At time t ($t > W$), the data from ACE_i in a sliding window block is expressed as a point cloud:

$$f_W(t) = \begin{bmatrix} x_{t-W+1}^1 & \cdots & x_{t-2}^1 & x_{t-1}^1 & x_t^1 \\ x_{t-W+1}^2 & \cdots & x_{t-2}^2 & x_{t-1}^2 & x_t^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{t-W+1}^N & \cdots & x_{t-2}^N & x_{t-1}^N & x_t^N \end{bmatrix} \in \mathbb{R}^{N \times W},$$

where t is time, the point cloud $f_W(t)$ is embedded in N -dimensional Euclidean space, each point in the point cloud $f_W(t)$ is a column $[x_j^1, x_j^2, \dots, x_j^N]^T$ of $f_W(t)$, and $j = t - W + 1, \dots, t$.

[0035] In some embodiments, pre-processing (i.e. data transformation) comprises using Takens' Embedding Theorem to convert the data to a point cloud. Consider a set of data points $\{x(t)\}$, observed at time values $t = 1, 2, \dots$. Instead of considering each point at time t in temporal isolation, the method comprises adding dimensions to the space. This incorporated the effect of looking back at D previous values of x , separated by time lags of length τ . In other words, the method involves replacing $x(i)$ by a D -dimensional vector of the form $\mathbf{v}_i = \{(i), x(i + \tau), \dots, x(i + D\tau)\}$, $i = 1, 2, \dots, T - D\tau$. The points $\mathbf{v}_i$ form a point cloud $\mathbf{v}$ (202) in the metric space. In the above matrix, $\mathbf{v} = f_W(t)$ can be viewed as a point cloud embedded in N -dimensional Euclidean space. Thus, the

data obtained at step 102 can be pre-processed into two or more, i.e. a plurality of, point clouds.

[0036] After generating the point cloud or point clouds, TDA feature extraction is performed (510). TDA is applied to extract one or more features for each data block in a sliding window. TDA extraction may comprise creating a persistence diagram to represent the TDA features for the point cloud and then using a persistence landscape to transform the TDA features in the persistence diagram into elements of a Hilbert space. Thus, with reference to Figures 2 and 5, a persistence diagram (204, 512) is first created. The persistence diagram captures the topological signature for each point cloud.

[0037] To generate a persistence diagram: for each data point $\mathbf{v}_i$, a radius of length $\epsilon \geq 0$ is defined around the data point. Each pair of points falling within that radius is then connected. Many connected components (called 0-dimension homology) or loops (1-dimension homology) are then identified. By gradually increasing ϵ , the corresponding homology changes. Some connections or loops disappear while new connections or loops appear.

[0038] Any two points will be connected if their Euclidean distance is less than ϵ . When $\epsilon = 0$, a connected component (i.e., each point in the point cloud $\mathbf{v}$) is born with birth $b = 0$. When gradually increasing ϵ to ϵ_1 , one point will become connected to other points. The original connected component (i.e., points connected now in $\mathbf{v}$) dies with death $d = \epsilon_1$. This gives us the first (b, d) pair and thus the first point $(0, \epsilon_1)$ in the persistence diagram appears. Recording the deaths for each point in $\mathbf{v}$ as ϵ continues to increase up to ϵ_q , and plot all the q corresponding (b, d) pairs to form the persistence diagram until all points are connected, where $q < W$. This process is applied to the point cloud $\mathbf{v}$ for each sliding window to build its persistence diagram 204.

[0039] Thus, by recording the corresponding ϵ of every birth (corresponding hole/loop appears) and death (corresponding hole/loop disappears) of a homology as $\{(b, d) \in \mathbb{R} \mid b, d \geq 0, b \leq d\}$, a persistence diagram is created comprising all points that persist within a predetermined as a radius for each data point changes. Thus, a different persistence diagram may be generated for each data point.

[0040] The structures that appear and disappear as radius ϵ is increased are called simplicial complexes (presently, a VietorisRips complex). These complexes are collections of simplices satisfying the condition that if a simplex is in the simplicial

complex, all of its subsets should also be in the complex. A k -simplex is a clique of $k + 1$ points, so a single point is a 0-simplex, two connected data points are a 1-simplex, and so on, for $k \in \mathbb{N}^+$. Given the point cloud $\mathbf{v}$, a persistent homology (e.g. 0-dimensional homology) is used to identify the topological features of the data.

[0041] The process of feeding topological information in a persistence diagram into a ML model is non-trivial. For this reason, the persistence diagram 204, generated at step 512, is converted into a sequence of piecewise-linear functions. The sequence of piecewise-linear functions may comprise a persistence landscape 206 (see Figure 2), generated at step 514 of Figure 5. Transforming the persistence diagram (at 512) with the captured TDA features to the persistence landscape (at 514) may thus comprise creating a respective landscape for each point in the persistence diagram, where the persistence landscape is a collection of functions of the respective landscape.

[0042] A persistence landscape provides a statistical summary of the topological features in data, using a real-valued function. For the persistence diagram 204, by connecting each of the points to the line $x = y$ using both horizontal and vertical lines, then rotating the space by 45 degrees clockwise, a landscape function 206 is obtained. The landscape is layered such that, when viewing parallel to the y -axis (vertical axis on the page) in the $-y$ direction, the landscape is layered such that a first layer comprises lines without obstruction (one is indicated with numeral 208 in Figure 2), a second layer comprises lines behind only one other layer (i.e. layer 210 behind only layer 208), a third layer comprises lines behind only two other layers (i.e. layer 212 behind both layers 208 and 210) and so on.

[0043] This process is further illustrated in Figure 6 in which a persistence diagram 600 is transformed into a persistence landscape 602. For each point $p = (b, d)$ in the persistence diagram 600, its landscape 602, $f_p(x)$, is defined as:

$$f_p(x) = \begin{cases} x - b, & b \leq x \leq \frac{b+d}{2} \\ d - x, & \frac{b+d}{2} < x \leq d \\ 0, & \text{otherwise} \end{cases}$$

[0044] A persistence landscape 602 is a collection of functions of $f_p(x)$, and is formally defined as:

$$\lambda_i(x) = \text{imax}_p f_p(x), i \in \mathbb{N}^+$$

where imax denotes the i^{th} largest value in $\{f_p(x) | p \in \{\text{persistence diagram}\}\}$. Each $\lambda_i(x)$ represents one feature vector. For example, lines 604, 606 and 608 represent the first, second and third landscapes in the persistent landscape, respectively.

[0045] For attack detection, a particular landscape is selected. For convenience, the first persistence landscape 604 is selected to illustrate TDA feature usage for attack detection.

[0046] Figure 7 shows examples of 0-dimensional persistence diagrams from three different sliding window blocks with $W = 25$, where the window contains no attack data (image (a)), partial attack data (image (b)), and all attack data (image (c)), respectively. The column of points for image (c) have a larger range than that for images (a) and (b). This implies that an attack can comparatively disperse columns (points). Next, persistence landscapes transform the persistence diagrams in images (a), (b) and (c) into functions.

[0047] Figure 7, image (d) shows the first landscape for 0-dimensional homology for points in images (a), (b) and (c). Different values exist on the y-axis for the three landscapes. This implies the feature pattern changes after the FDI attack is launched and thus can be captured by the ML algorithm to detect the anomaly. Moreover, in image (d), each landscape is discretized into a predetermined number of points (presently 100 points) with equal sampling intervals. Thus, per step 516 of Figure 5 – all points of the predetermined number of points for each landscape are the vectorized features from the sliding window for ACE signals, which will be fed into the ML algorithm or model 518.

[0048] The features in the feature vectors generated at 516 are fed into the ML algorithm 518 to construct a classification model, or to run or a previously trained classification model (520). The classification model is trained to classify each window as being indicative or an attack or not indicative of an attack. The ML results are then interpreted (by interpreter 522) to obtain a final detection output (524). The interpreter may give a positive result for a window if the window indicates an attack. However, as discussed below, the interpreter may not give a positive result for a window that indicates an attack, unless a pre-condition is met – e.g. there must be a predetermined number of consecutive positive results before the interpreter will give a positive result. Thus, the sensor data stream (502) that monitors the system's operations is converted to a Euclidean space point cloud (508) and then a TDA persistence diagram (512) is built to encode the data's topological features. The features encoded in the persistence diagram

(512) are then vectorized by turning the features into a TDA persistence landscape (514 then 516). Based on the vectorized features, a binary classifier is trained (520) to detect cyberattacks against AGC in power grids and other cyber physical systems.

[0049] To detect attacks from the vectorized features generated at 516, supervised ML (518) can be used to detect whether an attack is occurring in the grid. In some embodiments, the ML model is built based on a random forest (RF) algorithm. The RF algorithm is used to build a binary classification model 520 that takes the TDA features (516) as the input. A positive output from the model 520 indicates that an attack is underway. In other embodiments, the ML model may use a support vector machine (SVM) algorithm

[0050] The method may comprise producing one classification output per sliding window. This can be achieved by continually monitoring the time-sequence traces from the power grid with TDA-ML. That classification output indicates whether an attack is underway in that window. It is foreseeable that false alarms will arise. The method 100 may therefore further comprise waiting for a predetermined number of outputs (i.e. classified sliding windows) to indicate that an attack is occurring. Those outputs may be consecutive. While this will delay triggering an alarm in the event of an attacked, and reduce the time available for mitigation, awaiting a predetermined number of consecutive outputs to indicate an attack avoids raising false alarms – i.e. this process balances the false alarm rate (i.e., declaring that an attack is underway, when in fact there is not) and misdetection rates (i.e., not declaring that an attack is underway, when in fact one is). The predetermined number of outputs, or the length of each sliding window, may be selected based on the speed at which an attack, or a particular type of attack, can adversely affect the grid. For example, the method may involve waiting for $m = \mathbb{N}^+$ consecutive positive results before raising an alert to an operator of the generator for the grid, where m is called the fusion threshold – the number of consecutive results that is used to decide whether to raise an attack alert.

[0051] TDA-ML methods demonstrated herein, and the systems that implement them, significantly outperform other feature extraction methods in ML detectors, including the latest time-sequence feature extraction methods for the state-of-the-art FDI attacks. Moreover, when compared to the state-of-the-art DL alternatives, TDA-ML is more robust in dealing with different amounts of anomalous data and is much faster in attack detection.

Specifically, on average, TDA-ML can require at least 50% fewer AGC cycles to detect FDI attacks than LSTM, leaving significantly more time for real-time mitigation of attacks.

[0052] Figure 8 is a block diagram showing an exemplary computer device 800, in which embodiments of the invention – e.g. method 100 – may be practiced. The computer device 800 may be a mobile computer device such as a smart phone, a wearable device, a palm-top computer, and multimedia Internet enabled cellular telephones, an on-board computing system or any other computing system, a mobile device such as an iPhone™ manufactured by Apple™, Inc or one manufactured by LG™, HTCT™ and Samsung™, for example, or other device.

[0053] As shown, the mobile computer device 800 includes the following components in electronic communication via a bus 806 and in communication with external system over network 807:

- (a) a display 802;
- (b) non-volatile (non-transitory) memory 804;
- (c) random access memory ("RAM") 808;
- (d) N processing components 810;
- (e) a transceiver component 812 that includes N transceivers; and
- (f) user controls 814.

[0054] Although the components depicted in Figure 8 represent physical components, Figure 8 is not intended to be a hardware diagram. Thus, many of the components depicted in Figure 8 may be realized by common constructs or distributed among additional physical components. Moreover, it is certainly contemplated that other existing and yet-to-be developed physical components and architectures may be utilized to implement the functional components described with reference to Figure 8.

[0055] The display 802 generally operates to provide a presentation of content to a user, and may be realized by any of a variety of displays (e.g., CRT, LCD, HDMI, micro-projector and OLED displays).

[0056] In general, the non-volatile data storage 804 (also referred to as non-volatile memory) functions to store (e.g., persistently store) data and executable code. The system architecture may be implemented in memory 804, or by instructions stored in memory 804.

[0057] In some embodiments for example, the non-volatile memory 804 includes bootloader code, modem software, operating system code, file system code, and code to facilitate the implementation components, well known to those of ordinary skill in the art, which are not depicted nor described for simplicity.

[0058] In many implementations, the non-volatile memory 804 is realized by flash memory (e.g., NAND or ONENAND memory), but it is certainly contemplated that other memory types may be utilized as well. Although it may be possible to execute the code from the non-volatile memory 804, the executable code in the non-volatile memory 804 is typically loaded into RAM 808 and executed by one or more of the N processing components 810.

[0059] The N processing components 810 in connection with RAM 808 generally operate to execute the instructions stored in non-volatile memory 8104. As one of ordinary skill in the art will appreciate, the N processing components 810 may include a video processor, modem processor, DSP, graphics processing unit (GPU), and other processing components.

[0060] The transceiver component 812 includes N transceiver chains, which may be used for communicating with external devices via wireless networks. Each of the N transceiver chains may represent a transceiver associated with a particular communication scheme. For example, each transceiver may correspond to protocols that are specific to local area networks, cellular networks (e.g., a CDMA network, a GPRS network, a UMTS networks), and other types of communication networks.

[0061] The system 800 of Figure 8 may be connected to any appliance 818, such as AGC control system, a data store for a grid or other cyber physical system monitoring system, or an external database from which context can be acquired.

[0062] It should be recognized that Figure 8 is merely exemplary and in one or more exemplary embodiments, the functions described herein may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code encoded on a non-transitory computer-readable medium 804. Non-transitory computer-readable medium 804 includes both computer storage medium and communication medium including any medium that facilitates transfer of a computer program from one

place to another. A storage medium may be any available medium that can be accessed by a computer.

[0063] Throughout this specification and the claims which follow, unless the context requires otherwise, the word "comprise", and variations such as "comprises" and "comprising", will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

[0064] The scope of this disclosure encompasses all changes, substitutions, variations, alterations, and modifications to the example embodiments described or illustrated herein that a person having ordinary skill in the art would comprehend. The scope of this disclosure is not limited to the example embodiments described or illustrated herein. Moreover, although this disclosure describes and illustrates respective embodiments herein as including particular components, elements, feature, functions, operations, or steps, any of these embodiments may include any combination or permutation of any of the components, elements, features, functions, operations, or steps described or illustrated anywhere herein that a person having ordinary skill in the art would comprehend. Although this disclosure describes or illustrates particular embodiments as providing particular advantages, particular embodiments may provide none, some, or all of these advantages.

Claims

1. A method of detecting attacks against automatic generation control (AGC) of a grid, comprising:
 - obtaining data from the AGC;
 - pre-processing the data to generate a point cloud;
 - extracting topological data analysis (TDA) features from the point cloud;
 - inputting the TDA features to a machine learning (ML) model to generate a plurality of outputs; and
 - interpreting the outputs to determine whether the data includes an attack.

2. The method of claim 1, wherein the data is time-sequence signals.

3. The method of claim 1 or 2, wherein pre-processing the data to generate the point cloud comprises:
 - splitting the data into sliding window blocks of size W ; and
 - generating the point cloud $f_W(t)$ according to:

$$f_W(t) = \begin{bmatrix} x_{t-W+1}^1 & \cdots & x_{t-2}^1 & x_{t-1}^1 & x_t^1 \\ x_{t-W+1}^2 & \cdots & x_{t-2}^2 & x_{t-1}^2 & x_t^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{t-W+1}^N & \cdots & x_{t-2}^N & x_{t-1}^N & x_t^N \end{bmatrix} \in \mathbb{R}^{N \times W},$$

where t is time, the point cloud $f_W(t)$ is embedded in N -dimensional Euclidean space, each point in the point cloud $f_W(t)$ is a column $[x_j^1, x_j^2, \dots, x_j^N]^\top$ of $f_W(t)$, and $j = t - W + 1, \dots, t$.

4. The method of any one of claims 1 to 3, wherein extracting the TDA features from the point cloud comprises:
 - creating a persistence diagram to represent the TDA features for the point cloud; and
 - using a persistence landscape to transform the TDA features in the persistence diagram into elements of a Hilbert space.
5. The method of claim 4, wherein creating the persistence diagram to represent the TDA features for the point cloud comprises using 0-dimensional homology to capture the TDA features.

6. The method of claim 4 or 5, wherein transforming the persistence diagram with the captured TDA features to the persistence landscape comprises creating a respective landscape for each point in the persistence diagram, and wherein the persistence landscape is a collection of functions of the respective landscape.

7. The method of claim 6, wherein for each point $p = (b, d)$ in the persistence diagram, the respective landscape $f_p(x)$ is according to:

$$f_p(x) = \begin{cases} x - b, & b \leq x \leq \frac{b+d}{2} \\ d - x, & \frac{b+d}{2} < x \leq d \\ 0, & \text{otherwise} \end{cases}$$

8. The method of claim 7, wherein the persistence landscape is defined as:

$$\lambda_i(x) = \text{imax}_p f_p(x), i \in \mathbb{N}^+,$$

where imax denote the i -th largest value in the respective landscape $f_p(x)$.

9. The method of any one of claims 1 to 8, wherein the ML model is built based on random forest (RF) algorithm.

10. The method of any one of claims 3 to 9, wherein each output indicates whether a respective sliding window block indicates an attack is occurring, the method comprising generating alert upon detecting a predetermined number of consecutive outputs each representing that the respective data indicates an attack.

11. A system of detecting attacks against automatic generation control (AGC) of a grid, comprising a plurality of processors for:

obtaining data from the AGC;

pre-processing the data to generate a point cloud;

extracting topological data analysis (TDA) features from the point cloud;

inputting the TDA features to a machine learning (ML) model to generate a plurality of outputs; and

interpreting the outputs to determine whether the data includes an attack.

12. The system of claim 11, wherein the data is time-sequence signals.

13. The system of claim 11 or 12, wherein pre-processing the data to generate the point cloud comprises:

splitting the data into sliding window blocks of size W ; and

generating the point cloud $f_W(t)$ according to:

$$f_W(t) = \begin{bmatrix} x_{t-W+1}^1 & \cdots & x_{t-2}^1 & x_{t-1}^1 & x_t^1 \\ x_{t-W+1}^2 & \cdots & x_{t-2}^2 & x_{t-1}^2 & x_t^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{t-W+1}^N & \cdots & x_{t-2}^N & x_{t-1}^N & x_t^N \end{bmatrix} \in \mathbb{R}^{N \times W},$$

where t is time, the point cloud $f_W(t)$ is embedded in N -dimensional Euclidean space, each point in the point cloud $f_W(t)$ is a column $[x_j^1, x_j^2, \dots, x_j^N]^T$ of $f_W(t)$, and $j = t - W + 1, \dots, t$.

14. The system of any one of claims 11 to 13, wherein extracting the TDA features from the point cloud comprises:

creating a persistence diagram to represent the TDA features for the point cloud; and

using a persistence landscape to transform the TDA features in the persistence diagram into elements of a Hilbert space.

15. The system of claim 14, wherein creating the persistence diagram to represent the TDA features for the point cloud comprises using 0-dimensional homology to capture the TDA features.

16. The system of claim 14 or 15, wherein transforming the persistence diagram with the captured TDA features to the persistence landscape comprises creating a respective landscape for each point in the persistence diagram, and wherein the persistence landscape is a collection of functions of the respective landscape.

17. The system of claim 16, wherein for each point $p = (b, d)$ in the persistence diagram, the respective landscape $f_p(x)$ is according to:

$$f_p(x) = \begin{cases} x - b, & b \leq x \leq \frac{b+d}{2} \\ d - x, & \frac{b+d}{2} < x \leq d \\ 0, & \text{otherwise} \end{cases}$$

18. The system of claim 17, wherein the persistence landscape is defined as:

$$\lambda_i(x) = \text{imax}_p f_p(x), i \in \mathbb{N}^+,$$

where imax denote the i -th largest value in the respective landscape $f_p(x)$.

19. The system of any one of claims 11 to 18, wherein the ML model is built based on random forest (RF) algorithm.
20. The system of any one of claims 13 to 19, wherein each output indicates whether a respective sliding window block indicates an attack is occurring, the method comprising generating alert upon detecting a predetermined number of consecutive outputs each representing that the respective data indicates an attack.