Received December 10, 2021, accepted December 31, 2021, date of publication January 11, 2022, date of current version January 20, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3142022*

# Anomaly Detection for Insider Attacks From Untrusted Intelligent Electronic Devices in Substation Automation Systems

**XUELEI WANG[1], COLIN FIDGE[1], GHAVAMEDDIN NOURBAKHSH[2], (Member, IEEE), ERNEST FOO[3], (Member, IEEE), ZAHRA JADIDI[1,3], AND CALVIN LI[4]**

[1]School of Computer Science, Queensland University of Technology (QUT), Brisbane, QLD 4000, Australia
[2]School of Electrical Engineering and Robotics, Queensland University of Technology (QUT), Brisbane, QLD 4000, Australia
[3]School of Information and Communication Technology, Griffith University, Brisbane, QLD 4111, Australia
[4]Asset and Operations, Jemena Ltd., Sydney, NSW 2060, Australia

Corresponding author: Xuelei Wang (xuelei.wang@hdr.qut.edu.au)

This work was supported in part by the Cyber Security Research Centre Ltd., and in part by the Australian Government's Cooperative Research Centre's Program.

**ABSTRACT** In recent decades, cyber security issues in IEC 61850-compliant substation automation systems (SASs) have become growing concerns. Many researchers have developed various strategies to detect malicious behaviours of SASs during the system operational stage, such as anomaly-based detection. However, most existing anomaly-based detection methods identify an abnormal behaviour by checking every single network packet without any association. These traditional methods cannot effectively detect "stealthy" attacks which modify legitimate messages slightly while imitating patterns of benign behaviours. In this paper, we present feature selection and extraction methods to generalise and summarise critical features when detecting insider attacks triggering from untrusted control devices within SASs. By applying a sliding window-based sequential classification mechanism, our detection method can detect anomalies across multiple devices without the need to learn datasets collected from all devices. Firstly, to generalise critical features and summarise systems' behaviours so that it is unnecessary to collect all datasets, we selected and extracted six critical network features from generic object-oriented substation events (GOOSE) messages and seven summarised physical features based on the general architecture of the primary plant of distribution substations. After that, to improve detection accuracy and reduce computational costs, we applied sliding window algorithms to divide datasets into different overlapped window-based snippets. Then we applied a sequential classification model based on Bidirectional Long Short-Term Memory networks to train and test those datasets. As a result, our method can detect insider attacks across multiple devices accurately with a false-negative rate of less than 1%.

**INDEX TERMS** Intelligent electronic devices, substation automation systems, untrusted components, insider attacks, anomaly detection, sequential classification, sliding window.

## I. INTRODUCTION

In recent decades, cyber security issues in IEC 61850-compliant substation automation systems (SASs) have become growing concerns. Many researchers have developed various strategies to detect malicious behaviours of SASs during the system's operational stage. Some of them focused on knowledge-based detection by specifying legitimate rules and abnormal conditions based on expertise [1]–[4]. However, these knowledge-based methods can only detect known attacks [5]. Others targeted anomaly-based detection by applying machine learning algorithms to learn systematic behaviours [6]–[8]. Most existing anomaly-based detection methods identify an abnormal behaviour by checking every single network packet without any association between them. However, these traditional methods cannot effectively detect "stealthy" attacks which modify legitimate messages slightly while imitating patterns of benign behaviours. For instance, due to the special nature of substation operation, stealthy attacks which only alter the Boolean control signals from

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Raza.

"opening circuit breakers" to "closing circuit breakers" when a short circuit event happens, can still have severe consequences. Traditional methods cannot detect such attacks accurately as these attacks might be misclassified as benign behaviours. Thus, a bespoke anomaly-based detection model for detecting such attacks is required.

In this paper, we present feature selection and extraction methods, and sliding window-based sequential classification algorithms to detect "stealthy" attacks triggering from untrusted control devices within SASs. Our method targets such insider attack scenarios and can detect anomalies across multiple devices without the need to learn datasets collected from all devices. Both feature selection and extraction methods were applied as they both help generalise critical features and summarise systems' behaviours so that it is unnecessary to learn all datasets [6]. Additionally, sequential classification algorithms learn the contexts of the current behaviour, and thus, such additional information improves accuracy when detecting stealthy attacks [9]. Furthermore, sliding window algorithms divide the entire sequence of data into small snippets so that only recent contexts of the current behaviour are considered, and thus, improve detection accuracy and reduce computational costs [10].

The overall methodology in this paper: 1) collected datasets of both benign and malicious behaviours from a software-based simulation testbed and labelled datasets based on various behaviours; 2) selected and extracted critical network features from GOOSE messages and generic physical features from sensor data, and prepared datasets for machine learning; 3) used sliding window algorithms to divide datasets into different overlapped window-based snippets, and applied a sequential classification algorithm based on the bidirectional long short-term memory (BiLSTM) to train and test those datasets; 4) evaluated the approach by comparing experimental results of different anomaly-based detection methods, including decision tree and support vector machine algorithms; and 5) determined the preferred window size and step size in sliding window algorithms experimentally.

***Three main contributions*** in this paper are listed below:

- We present feature selection and extraction methods to generalise and summarise a total of 13 critical features when detecting stealthy insider attacks. Such methods help detect anomalies across multiple devices when only learning behaviours of one typical device.
- Compared to traditional detection methods, our detection algorithms combined a BiLSTM sequential classification algorithm and sliding window algorithms and improved detection accuracy by decreasing the false-negative rate from 30% to 1% approximately.
- Based on various experiments, we provide recommended settings for the window size and step size in sliding window algorithms for anomaly detection within SASs. The suggested settings balance the trade-off between detection accuracy and detection time.

## II. BACKGROUND

With the rapid advancement of information and communication technology, modern power grids have been experiencing a digitisation process during recent decades. Substation automation systems (SASs), also called the secondary plant, are critical components in power grids, that monitor and protect the primary plant (e.g., transformers) in substations. Legacy SASs involved numerous electro-mechanical components with intricate hardwiring in a centralised topology, significantly increasing operational complexity, configuration and maintenance costs, and potential safety hazards [11], [12]. Therefore, according to international standard IEC 61850 [13], new SASs have been developing continuously to satisfy contemporary high-level requirements regarding interoperability, maintainability, and flexibility [14].

However, IEC 61850-compliant SASs are vulnerable for various reasons. Firstly, when the IEC 61850 standard was first introduced, cyber security problems were not the main concerns as these issues were addressed later in another standard – IEC 62351 [15]. Nevertheless, many control devices from different vendors do not support IEC 62351 [16]. Secondly, as a new convenient feature for system administrators, remote access and control also increase risks of systems being penetrated since additional access portals are introduced [17]. Thirdly, the IEC 61850-compliant SASs support various communication protocols, including legacy protocols (DNP3, Modbus), and new protocols (MMS, GOOSE, SV). However, both legacy and new protocols are insecure due to improper authentication, lack of encryption, poor access control, and lack of integrity checks [18].

Lastly and importantly, protection relays, as a major component in SASs, also called intelligent electronic devices (IEDs), usually come from third-party vendors and may not be fully trusted by utility companies. According to utility companies' shared concerns, from the design specification stage to the deployment and operational stage, a control device (e.g., a protection relay) may become untrustworthy at any point. For instance, the device may acquire various vulnerabilities during the design and implementation stages. Although most vendors test and validate their new products before manufacturing, the assessment process may not be rigorous or standardised [19]. A vulnerable device is untrustworthy as it may become a weak point for attackers. Additionally, hidden malware and stealthy hardware Trojans may be introduced during the manufacturing process, either accidentally or deliberately [20]. Similarly, a validation engineer from a third-party supplier may install a hidden backdoor for future remote access which, even though introduced for altruistic reasons, could be exploited as part of an attack [21]. Finally, during system operation, random faults, misconfiguration or mistakes made during software or firmware upgrades can lead to a previously reliable device becoming untrustworthy.

Untrusted IEDs may stealthily perform harmful or unauthorised behaviours which could compromise or damage SASs, and thereby bring adverse impacts to the primary plant. A notorious incident was the BlackEnergy cyberattacks happened in Ukraine, where many IEDs had been compromised to send false "open the breaker" commands to numerous circuit breakers continuously, and it caused a blackout for a broad area [22]. Thus, it is important to detect abnormal behaviours from an untrusted IED before it brings about catastrophic consequences. According to asset management standard ISO 55000 [23], it is preferred to examine and exclude such devices before they enter an operational environment. However, the effort involved in independently evaluating every device to be installed in a complex system such as a substation makes this impractical [24]. Therefore, common alternative techniques focus on monitoring and detecting untrustworthy behaviours during system operation, following the installation of malicious devices. During system operation, anomaly-based intrusion detection techniques are suitable to detect anomalies in time-critical communication as they only bring minimal side-effects to normal system operations while can also detect zero-day attacks [5].

Furthermore, according to IEC 62351 [15], various types of attacks are based on four types of threats that compromise four fundamental security requirements – confidentiality, integrity, availability, and non-repudiation. Since availability is the most essential requirement in power systems, most SASs have both main (X) systems and backup (Y) systems to provide additional guarantees for critical devices and communication paths. This countermeasure mitigates the impacts of denial of service (DoS) attacks. On the other hand, false data injection attacks (FDIA) violate the integrity requirement, and can also bring severe impacts even by only altering the Boolean control signals from "open circuit breakers" to "close circuit breakers" when a short circuit event happens [4]. Additionally, with various "cloaking" strategies, including not triggering in system testing mode, only modifying message payloads slightly, and imitating patterns of benign behaviours, these "insider" attacks are "smart" enough to avoid being identified by traditional detection mechanisms [8]. Therefore, insider attacks, including FDIA and replay attacks, are our main concerns when protecting power systems [25]. Generic Object Oriented Substation Event (GOOSE), as an essential communication protocol within SASs, transmits time-critical protection and control signals such as alarms, trips, and interlocks among IEDs [26]. Thus, this paper mainly focuses on insider attacks targeting GOOSE messages.

According to the principle of anomaly-based detection [27], two factors determine detection performance regarding time efficiency and accuracy: 1) the selection of features from datasets, and 2) the choice of machine learning algorithms. According to different proprietary communication protocols within power systems, various specific features are required to detect different threat scenarios. Meanwhile, if selected features are redundant, it will increase the diagnosis time when detecting anomalies, and thus add communication latency to normal system operation. Conversely, if the selected features are not sufficient, detection accuracy will decrease. Even worse, with insufficient features, it may not be possible to detect "stealthy" attacks as the altering features may be ignored. Furthermore, different machine learning algorithms are suitable for various application scenarios, and it is important to select appropriate ones for the best performance.

## III. RELATED WORK

This section contains a comprehensive literature review. Since GOOSE is the most critical communication protocol within SASs, we firstly describe several network features of GOOSE messages. Then, we introduce physical features from sensor data, and review the possibility of combining physical features and network features to detect anomalies within SASs. After that, we point out the issue of promoting the utilisation of limited datasets into general and systematic problems, and provide potential feature extraction methods to overcome such issues. Lastly, four typical machine learning algorithms for anomaly detection are reviewed. We also summarise related works of applying sliding window algorithms to improve the accuracy of machine learning classification in various applications.

### *A. NETWORK FEATURES OF GOOSE*

There are two types of proprietary features of GOOSE – dynamic features and static features [28]. Dynamic features are usually calculated based on the statistical trends of traffic volume and traffic frequency [28]. Kwon *et al.* [28] defined three particular features relating to GOOSE messages based on RFM analysis in business and marketing research. They defined the last GOOSE arrival time as Recency, the mean time interval of GOOSE arrival time (also known as the heartbeat) as Frequency, and the total GOOSE arrival count as Monetary. On the other hand, static features are often filtered and extracted from different fields in a single GOOSE packet [29]. Based on the standardised GOOSE structure defined in the IEC 61850-8-1 standard [30], static features of GOOSE include MAC address, "APPID", "gocbRef", "stNum", "sqNum", Boolean control signals from the "allData" field, etc.

Many researchers have applied both dynamic features and static features to monitor and identify abnormal GOOSE messages in SASs [1], [3], [4], [28], [29], [31], [32]. However, some of them failed to provide detailed statistical results, such as a false-positive rate (FPR) and false-negative rate (FNR) [28], [29] while the others did not consider or failed to detect stealthy attacks [1], [3], [4], [32]. Therefore, more features are required to improve the accuracy of detecting stealthy attacks.

### *B. PHYSICAL FEATURES FROM SENSOR DATA*

From an electrical engineering perspective, based on Kirchhoff current and voltage laws, Valdes *et al.* [7] used sensor data, such as current and voltage, to classify three distinct states corresponding to normal operation, non-malicious fault, and false measurement injection in SASs. However, their approach is limited to detecting attacks on SV messages and the accuracy is not satisfactory. Additionally, Kreimel *et al.* [8] target communication among Remote Terminal Units (RTUs) based on the DNP3 protocol. They selected both dynamic features (round-trip-time of packets) and static features (packet length, TCP window size) as well as sensor data (voltage) collected from solar panels. They achieved high detection accuracy on man-in-the-middle (MITM) drop attacks, but low accuracy on stealthy attacks which only change the transmitted measurement data by a small amount. This low accuracy can be attributed to the fact that the FDIA did not deviate much from the values of normal behaviour [8]. As a result, we conjectured that a similar method can be applied to detect stealthy attacks targeting GOOSE messages by carefully selecting both critical network features from GOOSE messages and generic physical features from sensor data.

## C. COMBINING BOTH NETWORK AND PHYSICAL FEATURES

In our previous work [33], we identified and selected one dynamic feature (the GOOSE heartbeat), nine static features (e.g., MAC, APPID, gocbRef, allData), and two types of physical features (circuit physical values and circuit breaker status). By including two additional features, Boolean control data from the ''allData'' field and various physical features, our method improved the accuracy of detecting stealthy attacks by decreasing the false-negative rate from 25% to 5% approximately.

However, we also observed an issue when introducing physical features to anomaly detection in SASs. Within SASs, multiple instances of the same type of IEDs may be applied to protect different sections of the primary plant. When the same types of IEDs are compromised, IEDs protecting different parts will generate datasets with different network and physical features, for instance, when IED1 provides overcurrent protection to transformer1 while IED2 protects transformer2. When both compromised devices IED1 and IED2 are triggered to conduct the same FDIA respectively, network features generated from both IED1 and IED2 may have the same patterns. However, physical features generated from sensor data are different as two attacks impact different physical parts. The attack from IED1 only influences physical values and statuses around transformer1 without interfering with transformer2 while the attack from IED2 is totally opposite. Therefore, even though IED1 and IED2 are the same types of devices, if an anomaly-based detection model which applies both network features and physical features only learns attack datasets from IED1, it cannot detect attacks from IED2 accurately. Due to this reason, the detection model must learn all attack datasets generated from all devices with an SAS. However, since SASs are complex systems that involve numerous control devices, it is impractical to collect attack datasets which contain anomalies occurring on every single device [34]. Thus, an additional feature extraction method is required to generalise and summarise critical features to detect anomalies across multiple devices while only learning behaviours of one typical device [35].

## D. FEATURE EXTRACTION

Some researchers have proposed several feature extraction methods, and those methods might be useful to overcome the issue mentioned above Ouyang *et al.* [6] presented a hierarchical time series feature extraction method to detect anomalies in power consumption. They defined four types of features from daily power consumption readings – summary features, shift features, transform features, and decompose features. The summary features are time-windowed statistical variables, including mean, median, and standard deviation of daily power consumption. Qiu *et al.* [36] also introduced trend indicators to detect anomalies for power consumption. The trend indicators are calculated based on the average values of the time series. Although these features are based on daily power consumption, a similar method can be applied to summarise physical features in our previous method [33] to detect anomalies across multiple devices. Furthermore, Gomes *et al.* [37] summarised two general feature extraction methods for streaming data – summarisation sketches and dimensionality reduction. Summarisation sketches combine any sketches of individual streams in a space-efficient way while dimensionality reduction converts original input data into a simplified form without compromising relevant patterns of the input data. These two methods can also be helpful to promote the utilisation of limited datasets into general and systematic problems.

## E. TYPICAL MACHINE LEARNING ALGORITHMS FOR ANOMALY DETECTION

Generally, when applying machine learning algorithms for anomaly detection, there are two main types: supervised learning and unsupervised learning. Although unsupervised clustering algorithms do not require upfront effort to label datasets appropriately, their detection accuracy is usually lower than supervised classification algorithms [38]. For supervised learning, according to different classification outputs, there are commonly two types: 1) classifying each sample with one label, and 2) classifying a consecutive sequence of samples with one label. The former one usually applies traditional algorithms, such as K-Nearest Neighbour (KNN), Support Vector Machine (SVM), and Decision Tree, while the latter one adopts sequential classification algorithms, including Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM).

The KNN algorithms classify a new abnormal behaviour by a majority vote from its near neighbours [37]. The SVM algorithms find a hyperplane to clearly distinguish data points into two groups based on various features [10]. The decision tree algorithms improve the accuracy of classification through various abnormal factors in a tree structure [39]. These traditional algorithms usually do not consider time-serial correlations among various features, and may not accurately detect stealthy attacks in which messages are only modified marginally [8].

On the other hand, sequential classification algorithms, such as LSTM, have feedback connections which can learn the contexts of the current behaviour, and thus provide more accurate results when detecting stealthy attacks [9]. Meanwhile, LSTM is better for a short time duration of time-serial data and can reduce the system complexity [9]. Since most transient behaviours in substations are short-term events with tangled logical processes, LSTM is suitable for identifying such systematic behaviours. Furthermore, bidirectional LSTM (BiLSTM) models are evolved from LSTM, which consists of two additional layers: a forward LSTM layer, and a backward LSTM layer. By learning both the forward flow and the backward flow, BiLSTM can effectively understand the correlation patterns among the previous behaviour, the current behaviour, and the next behaviour [40].

### F. SLIDING WINDOW ALGORITHMS

In recent decades, sliding window algorithms have become popular when learning time-series streaming data. They can effectively reduce learning computational costs and improve detection accuracy when detecting anomalies from time-series streaming data [34]. The sliding window is defined as a sequence of data which represents the most recently arrived tuples [41]. It covers W number of samples and pushes forward S number of samples every window. W is called the window size while S is called the step size. Generally, there are two types of sliding windows – quantity-based (count-based) and time-based [34], [42]. Quantity-based defines the window size based on a number of samples while time-based specifies the window size based on a period of time.

Many researchers have applied sliding window algorithms to various applications. In the field of power systems, some researchers utilised sliding window algorithms for power supply load forecasting [43], transient stability prediction [44], faulty equipment detection based on image recognition [45], and IED defect classification based on text mining [40]. In the field of anomaly detection, researchers applied sliding window algorithms to detect anomalies in different applications, such as IoT networks [9], [10], [46], and in-vehicle networks [47], [48]. However, none of them has applied sliding window algorithms to detect anomalies within SASs.

Furthermore, according to research from both fields, it is widely believed that sliding window size is an important factor in sliding window algorithms, and it is application-specific to select an appropriate sliding window size. In the field of anomaly detection, researchers also emphasised that there is a trade-off between the detection accuracy and detection time when choosing different sliding window sizes [49]. The larger the window size, the higher the accuracy, the more the detection time (the time between the start of the attack and the attack is detected), and vice versa [34], [47], [48]. Meanwhile, however, there is no empirical procedure or standard which can help us determine a preferred window size in the application field of anomaly detection. Therefore, the preferred window size must be selected based on security requirements considering both accuracy and detection time.

## IV. DATASET COLLECTION

In our research, the datasets were collected from a simulation testbed. Although datasets generated from real operational systems are better than simulated datasets, such datasets usually lack attack scenarios or do not indicate if they contain malicious behaviours [11]. Furthermore, datasets for commercially sensitive critical infrastructure such as SASs are difficult to obtain. On the other hand, with a certain level of fidelity, simulation testbeds can generate more varieties of datasets, including both benign behaviours and various types of malicious behaviours. Thus, in our work, based on the IEC 61850 standard, a cost-efficient software-based simulation testbed was implemented. We generated and collected a total of 31 datasets based on various scenarios, including 15 benign scenarios and 16 attack scenarios.

### A. SIMULATION TESTBED

The testbed[1] runs on Oracle VirtualBox with five virtual machines (VMs). One VM simulates a small-scale primary plant of a distribution substation using MATLAB/Simulink[2]. According to the general architecture of distribution substations, the simulated primary plant consists of a 66kV high-voltage line, two transformers, a 22kV low-voltage line, four feeders, and many circuit breakers (shown in Figure 1). Ten short-circuit fault blocks were set up in the Simulink for generating non-malicious events at ten different locations. The rest of the four VMs represent different types of IEDs simulated using OpenPLC[3], including three instantaneous overcurrent protection – IED_PIOC_TRSF1 (IED1), IED_PIOC_TRSF2 (IED2), and IED_PIOC_FDR and one circuit breaker failure protection – IED_BFP. Communication networks among each VM, such as GOOSE trip messages between IEDs and the primary plant, were written in C/C++ according to libiec61850[4]. Additionally, in each VM, various interface programs were written to link OpenPLC, MATLAB/Simulink, and the "libiec61850" library. As shown in Figure 2, the interface program in VM-IEDs reads analogue values from Simulink in VM-Primary-Plant via UDP packets, and passes these values to OpenPLC. Meanwhile, the program also reads digital signals from OpenPLC and passes these signals to the "libiec61850" program to assemble GOOSE packets. After the "libiec61850" program in VM-Primary- Plant receives those GOOSE packets, the interface program reads digital signals from decoded packets, and passes them to Simulink via UDP packets. Although the testbed only includes a limited number of IEDs as a prototype,

[1]Testbed implementation details are at https://github.com/kaitoray/Stage2
[2]https://www.mathworks.com/products/simulink
[3]https://www.openplcproject.com
[4]https://libiec61850.com/libiec61850/

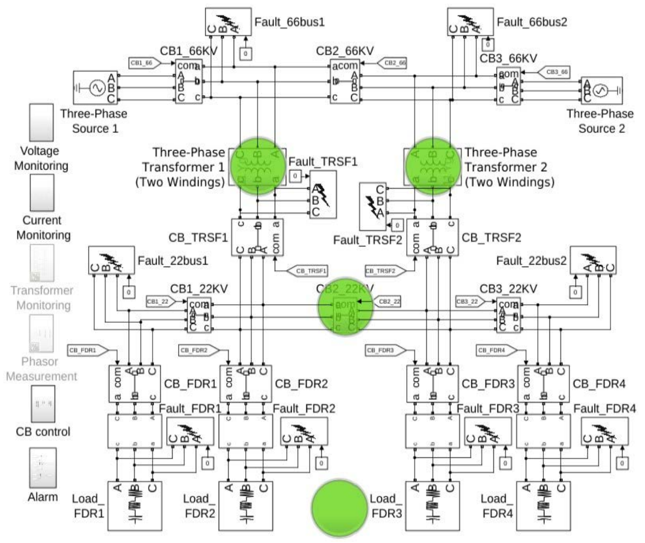

**FIGURE 1.** The primary plant simulated in MATLAB/Simulink.

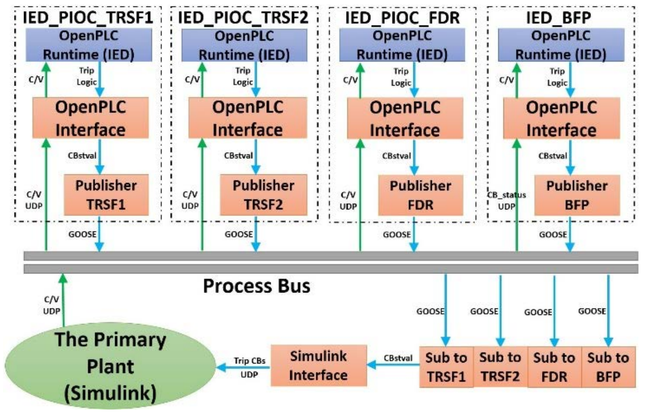

**FIGURE 2.** The architecture of testbed communication networks.

this can be extended with more functionalities in the future. Figure 2 illustrates the communication architecture of the testbed. In particular, the central process bus is used by IEDs to communicate with actuators (e.g., circuit breakers) in the primary plant via GOOSE messages.

### B. DATASET GENERATION

Generally, there are two different benign behaviours in substation operation: 1) normal operation when no unusual events happen; and 2) emergency operation when non-malicious events (e.g., short-circuit faults) happen. Attacks can occur in both benign behaviours, which introduces two types of malicious behaviours: 1) an attack under normal operation to disrupt energy transmission, and 2) an attack under emergency operation to stop protection mechanisms or trigger undesirable protection operations.

Based on these four types of behaviours, a total of 31 datasets[5] were generated. Each dataset consists of five network packet capture files collected from five VMs, and four sensor data records collected from four IEDs. Since the testbed only models instantaneous overcurrent protection and circuit breaker failure protection, the sensor data only contains circuit current values from sensors at different locations and the operational status of various circuit breakers. Table 1 describes each scenario. Datasets of both normal operation and emergency operation are benign behaviour datasets which contain 7447 and 12457 individual samples respectively. Attack datasets include attack scenarios from both IED1 and IED2 with 8015 and 9902 individual samples respectively. All benign behaviour datasets and attack datasets of IED1 were used for training only while attack datasets of IED2 were used for testing only.

As mentioned before, this paper mainly focuses on FDIA and replay attacks. We created eight different attack scenarios regarding GOOSE messages from IED1 and replicated these eight scenarios to GOOSE messages from IED2. IED1 and IED2 are the same type of protection relays that protect transformer1 and transformer2 respectively. All these eight attack scenarios can be classified as FDIA. According to Ahmed and Pathan [50], FDIA consists of three forms in general: 1) deletion of data from the original message; 2) modification of data in the original message, and 3) addition of fake data or fake messages. We cover the last two forms in this paper and implemented four attack scenarios for both the normal operation and the emergency operation respectively. For the normal operation, four attack scenarios include: 1) two message injection attacks that inject additional GOOSE trip messages to mislead circuit breakers into opening; and 2) two message modification attacks that change the payload of GOOSE messages from non-trip to trip to mislead circuit breakers into opening. For the emergency operation when a phase-to-phase fault happens, four attack scenarios contain: 1) two message injection attacks that a) inject additional GOOSE non-trip messages to stop protection mechanisms, and b) inject additional GOOSE trip messages of another IED to trigger unnecessary and unexpected protection mechanism; and 2) two message modification attacks that a) change the payload of GOOSE messages from trip to non-trip to stop the protection mechanism, and b) change the payload of GOOSE messages of another IED from non-trip to trip to trigger unnecessary and unexpected protection mechanisms.

## V. DATASET PRE-PROCESSING

After collecting all the datasets, we conducted a comprehensive dataset pre-process, and prepared various datasets for different controlled trials later. Firstly, we handled all datasets with three processes – format conversion, data merging, and data normalisation. Then we selected nine critical distinguishing features from GOOSE messages and various physical features (circuit current values and circuit breaker statuses) from sensor data needed for precisely anomaly detection of insider attacks. Meanwhile, we applied a feature extraction method to generalise and summarise

[5]Datasets are available at https://dx.doi.org/10.21227/qr8j-h344

**TABLE 1. Generated datasets of different scenarios.**

| Type of Behaviours | # of scenarios | # of individual samples | Labels | Scenarios | Description |
|---|---|---|---|---|---|
| Normal operation | 1 | 7447 | 0 | No events | No unusual events happen |
| Emergency operation | 10 | 8563 | 1 | Overcurrent protection | A phase-to-phase fault happens, associated overcurrent protection triggered |
| | 4 | 3894 | 1 | Breaker failure protection | Overcurrent protection failed; breaker failure protection triggered |
| Attacks from IED1 under normal operation | 2 | 1253+1385 | 901, 903 | Replay | IED1 injects replayed GOOSE trip messages |
| | 2 | 1093+1294 | 902, 904 | FDIA | Original non-trip messages from IED1 are modified to trip messages |
| Attacks from IED1 under emergency operation | 2 | 864+730 | 905, 907 | Replay | IED1 injects replayed legitimate messages to stop protection or trigger unexpected protection |
| | 2 | 811+585 | 906, 908 | FDIA | IED1 modifies original messages to stop protection or trigger unexpected protection |
| Attacks from IED2 under normal operation | 2 | 1930+1624 | 901, 903 | Replay | IED2 injects replayed GOOSE trip messages |
| | 2 | 1964+1552 | 902, 904 | FDIA | Original non-trip messages from IED2 are modified to trip messages |
| Attacks from IED2 under emergency operation | 2 | 870+654 | 905, 907 | Replay | IED2 injects replayed legitimate messages to stop protection or trigger unexpected protection |
| | 2 | 772+536 | 906, 908 | FDIA | IED2 modifies original messages to stop protection or trigger unexpected protection |

critical features from both network and physical features. Six network features and seven summarised physical features were extracted. After that, we labelled each sample based on its behaviours. Since sequential classification algorithms classify a sequence of samples with one label, we applied the worst-case principle to generate labels for each sequence. Lastly, we applied sliding window algorithms to divide datasets into different overlapped window-based snippets. Both the non-window-based datasets and window-based datasets were generated to satisfy traditional machine learning algorithms and sequential classification machine learning algorithms respectively.

## A. DATA HANDLING PROCESSES

Before selecting critical features from datasets, three necessary dataset handling processes were required. These include format conversion, data merging, and data normalisation, which are demonstrated below. Some processes are proprietary to the simulation environment in this paper and may require slight adjustment to apply to other cases.

### 1) FORMAT CONVERSION

Since original datasets contain network packet capture (PCAP) files which cannot be directly used, we firstly converted network packet files to comma-separated values (CSV) files. As stealthy attacks targeting GOOSE messages are our main concern, only GOOSE packets were extracted. We wrote a Python program using Scapy to elicit all static features from the GOOSE packets as well as the packet received timestamp, and exported this data to CSV files.

### 2) DATA MERGING

In this step, we merge all CSV files into one CSV file and remove redundant data. Firstly, we merged all converted network CSV files into one file and removed redundant network packets. Then, to link packet transmission events to physical sensor data, according to the packet received timestamp of each packet, we wrote a macro to find the closest timestamp from four sensor data records, and added corresponding physical features after that packet. Finally, we generated one CSV file which contains both network features and physical features.

### 3) DATA NORMALISATION

Some data may be missing or invalid and some data may be difficult to recognise by machine learning algorithms, thus, requiring a data normalisation process. Firstly, we removed all missing and invalid data. Secondly, all non-numerical values were converted to unique numerical values to reduce computational costs. For instance, the MAC address "20:17:01:16:F0:99" was simplified to 99, gocbRef "Testbed/PIOC$TRSF1$CBStval" was changed to 1111, and Boolean control value [1, 1, 1, 0] in the "allData" field was converted to a binary number "1110" first, then converted to the corresponding decimal number "14".

## B. FEATURE SELECTION

After finishing data handling processes, we selected nine critical network features and two genres of physical features, which are demonstrated below. All network features were selected to detect various specific attack scenarios, especially for stealthy attacks including FDIA and replay attacks. Two types of physical features were selected to indicate physical systems' behaviours, and accordingly, help determine network behaviours.

#### 1) HEARTBEAT (NETWORK)

An important dynamic feature in GOOSE message communications. For packets with the same "gocbRef", we calculated the time intervals of any two near packets as the heartbeat. This dynamic feature normally indicates if non-malicious events happen, but also can be used to detect DoS attacks, FDIA, and time delay attacks [2].

#### 2) MAC-SRC (NETWORK)

The source MAC address, which can be used to identify the publisher to detect MITM attacks and FDIA [1].

#### 3) APPID (NETWORK)

GOOSE application identification, which can be used to verify the type of application and detect if an illegal application is sent from the publisher.

#### 4) LENGTH (NETWORK)

The length of the GOOSE header and APDU, which can be used to detect if a GOOSE message is invalid or modified.

#### 5) gocbRef (NETWORK)

The GOOSE control block reference, which contains all information of a pre-defined control block. Since the "datset" field and the "goID" field are included in the "gocbRef" field, only the "gocbRef" field was selected.

#### 6) DIF-ST (NETWORK)

The differential value of the state number. For packets with the same "gocbRef", we calculated the differential value between the current stNum and the previous stNum. This normally indicates if non-malicious events happen, but also can be used to detect FDIA by highlighting large jumps in the number sequence [4].

#### 7) DIF-SQ (NETWORK)

The differential value of the sequence. Similar to Dif-st.

#### 8) numDatSetEntries (NETWORK)

Indicates the amount of data in the "allData" field and can be used to detect if additional data is included in payloads.

#### 9) DEC-allData (NETWORK)

The decimal number of converting all Boolean values in the "allData" field, which normally invokes the control commands, and can be used to detect FDIA targeting Boolean control value.

#### 10) CIRCUIT PHYSICAL VALUES (PHYSICAL)

Normally consists of current and voltage readings observed from various sensors, which imply various power systems' behaviours. By checking if the circuit current values are normal or not, it helps distinguish between malicious behaviours (e.g., the relay was attacked to make circuit breakers open) and benign behaviours (an actual short-circuit fault occurred).

#### 11) CIRCUIT BREAKER STATUSES (PHYSICAL)

The statuses of different circuit breakers, usually collected from system logs in each IED, also help determine systems' behaviours.

### *C. FEATURE EXTRACTION*

In practice, it is costly and time-consuming to generate all attack datasets targeting each individual IED as there are many IEDs on the process bus of an IEC 61850 compliant SAS. As we mentioned before in the literature review, the anomalous behaviours of one IED may be different from another one, even if they are the same type of devices. These differences reflect on both network features (e.g., MAC address and APPID) and physical features (e.g., current values around transformer1 and transformer2). Therefore, it is important to generalise our detection methods to be suitable for general cases. We need to detect anomalies from all IEDs in a small-scale simulation environment while only learning datasets from one typical IED. Accordingly, with minor adjustments, our methods can be extended and applied to large-scale real systems. To achieve this objective, in this work we applied feature extraction methods to generalise critical features from both network features and physical features.

Firstly, we excluded three network features which indicate the identity of a particular device or a particular message application. These three network features include MAC-src, APPID, and gocbRef. Without such identities, the network behaviours of an insider attack are generalised to arbitrary devices, and thus, the detection model only needs to learn malicious behaviours from one typical device. However, without such identities, we can only detect if there is a malicious message, but cannot identify where this message comes from directly. Nonetheless, after detecting the malicious message, we can still re-extract such identities from the message payload, and trace the sources of the anomaly.

Secondly, we summarised various physical readings and reduced the number of physical features from 18 to 7. Although there are only two types of physical features, various sensors and system logs still generate numerous features which indicate the physical statuses of different zones within a substation. Based on the general architecture of the primary plant of distribution substations, we extracted six summarised features based on various circuit physical readings. Each feature is the average value of one horizontal level of the primary plant as shown in Figure 3. If one of these physical features is abnormal, it means there is something wrong at that horizontal level, and accordingly, indicates the corresponding IEDs which protect that level may be compromised to launch attacks. Furthermore, we also summarised all circuit breaker statuses into one feature. Since most circuit breakers only

**TABLE 2.** Different labelling methods.

| Sequence of behaviours | Source of packets | Label type 1 | Label type 2 | Label type 3 | Sequential Labelling | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Normal** | IED1 | 0 | 0 | 0 | 0 | | | | | | |
| **Normal** | IED2 | 0 | 0 | 0 | 0 | 1 | 1 | | | | |
| **Normal** | IED3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | |
| **Emergency (IED1 response)** | IED1 | 1 | 1 | 1 | | 1 | 1 | 1 | | | |
| **Emergency (IED1 response)** | IED2 | 0 | 1 | 1 | | | 1 | 1 | 901-908 | | |
| **Emergency (IED1 response)** | IED3 | 0 | 1 | 1 | | | | 1 | 901-908 | 901-908 | |
| **Attack on IED1** | IED1 | 901-908 | 901-908 | 901-908 | | | | | 901-908 | 901-908 | 901-908 |
| **Attack on IED1** | IED2 | 0 | 901-908 | 0 or 1 | | | | | | 901-908 | 901-908 |
| **Attack on IED1** | IED3 | 0 | 901-908 | 0 or 1 | | | | | | | 901-908 |

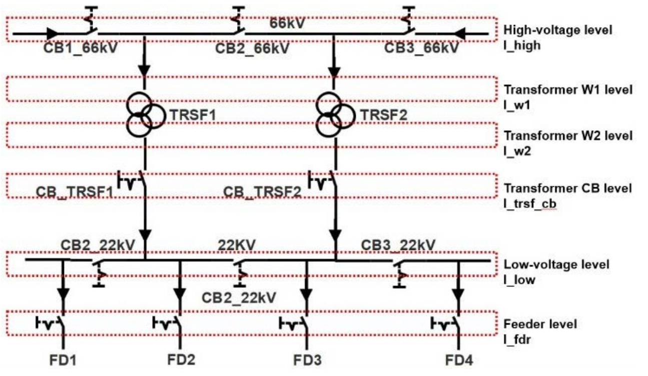

**FIGURE 3.** Summarised physical features based on different "horizontal" levels of the primary plant.

have two statuses – open (1) and closed (0), we compiled all circuit breaker statuses following a specific sequence and showed them as a binary number.

As a result, we generalised and extracted a total of 13 features which are listed below. With a limited volume of datasets that only contain malicious behaviours from one typical IED, these 13 features can help detect anomalies from all IEDs. Additionally, reducing the number of features also helps simplify the complexity of the training neural network, accordingly, and decreases training and detection time while increasing detection accuracy.

**Six critical network features:**

***1) GOOSE Heartbeat***

***2) Length of GOOSE packet***

***3) Dif-st***

***4) Dif-sq***

***5) numDatSetEntries***

***6) Dec-allData***

**Seven summarised physical features:**

*1) I-high (physical):* the average current values among high-voltage level

*2) I-w1 (physical):* the average current value among transformers' winding 1 (near high-voltage side)

*3) I-w2 (physical):* the average current value among transformers' winding 2 (near low-voltage side)

*4) I-trsf-cb (physical):* the average current value among all transformers' circuit breakers (near low-voltage side)

*5) I-low (physical):* the average current value among low-voltage level

*6) I-fdr (physical):* the average current among all feeders

*7) Bin-cb-status (physical):* a binary sequence of statuses of all circuit breakers.

## *D. LABELLING*

According to the various scenarios shown in Table 1, different labels were given. Label 0 indicates the normal operation scenario when no unusual event happens. Label 1 shows the emergency operation scenario when a non-malicious event happens. Labels 901 – 908 represent different stealthy attack scenarios. According to the characteristic of publisher-to-subscriber communication, GOOSE network traffic involves repeated network packets publishing from different IEDs. Since our datasets contain both network features and physical features, there are three types of labelling methods which are illustrated in Table 2.

Label type 1 focuses on network features. Only the packet from the impacted IED is labelled as an emergency or attack, and the rest are labelled as normal. Label type 2 emphasises physical features, labelling all packets as 0 under normal operation, 1 under emergency operation, and 901 to 908 under various attacks. Label type 3 is the combination of type 1 and type 2 which was used in this paper. Under normal operation, all packets are labelled as 0. Under emergency operation, all packets are labelled as 1. Under attacks, only the packet from the impacted IED is labelled as an attack, and the rest of the packets are labelled as emergency operations due to abnormal physical features.

Datasets for sequential classification algorithms require different labelling methods as such algorithms classify a sequence of samples with one label. Therefore, within a sequence of samples, different labels need to be integrated into one label. According to Baldini [48], they labelled a sequence of samples as attacks if it contains at least one malicious packet. We improved their labelling methods and applied the worst-case principle that labelling a sequence of samples based on the highest-priority label (attacks higher than emergency, emergency higher than normal). Table 2 illustrates examples of labelling a sequence. If a sequence of samples involves any attack labels (label 901 to 908), it will be labelled as the corresponding attack. If a sequence of samples does not contain attack labels, but has label 1, it will be labelled as 1. Only a sequence of samples with all labels 0 will be labelled as normal.

**Algorithm 1** The Quantity-Based Sliding Window

```
1: Input: Entire sequence of samples $X = \{X_1, X_2, X_3, \dots, X_n\}$, window size = w packets, step size = s packets
2: Output: Different overlapped window-based snippets $Y = \{Y_1, Y_2, Y_3, \dots, Y_m\}$
3: $k = 1$
4: $m = \lfloor (n - w) / s \rfloor + 1$
5: for $j = 1$ to m do
6:   $Y_j = \{X_k, X_{k+1}, X_{k+2}, \dots, X_{k+w-1}\}$
7:     $k = k + s$
8: end for
```

## E. SLIDING WINDOW PROCESS

In this step, we divided the whole sequence of samples into different overlapped window-based snippets. We applied both the quantity-based and the time-based sliding window algorithms.

For the quantity-based sliding window algorithm (shown in Algorithm 1), each window-based snippet has the same number of packets. The window size is set to w packets while the step size is set to s packets. Firstly, we extracted the first snippet with w number of packets from $X_1$ to $X_w$. Then we slid the window forward s packets and get the second snippet with a size of w packets from $X_{1+s}$ to $X_{1+s+w}$. After that, we repeated the process until we reached the last packet.

For the time-based sliding window algorithm (shown in Algorithm 2), each snippet has the same time interval. The window size is set to w seconds while the step size is set to s seconds. Similar to the quantity-based algorithm, we extracted the first window by finding the first w seconds of packets, then step forward s seconds every time to get the rest of the windows until reading the last packet.

## F. FINAL DATASET SUMMARY

After processing all the datasets, we generated a total of six groups of datasets for later evaluation which are shown in Table 3. In each group, we divided a total of 31 datasets into training datasets and testing datasets. Training datasets contain 15 benign scenarios (one normal and 14 emergency) and eight insider attack scenarios triggered from IED1. Testing datasets include eight insider attack scenarios triggered from IED2 which were never used for training purposes. The training datasets involves a total of 27919 individual samples, while the testing datasets have 9902 individual samples. For window-based datasets, they still have the same number of individual samples, but have different amounts of window-based samples according to different window sizes and step sizes. For instance, if the window size is 8 packets and the step size is 1 packet, the number of training window-based samples is 27912. If the window size is 8 packets and the step size is 2 packets, the number of samples is 27912 / 2 = 13956.

**Algorithm 2** The Time-Based Sliding Window

```
1: Input: Entire sequence of samples $X = \{X_1, X_2, X_3, \dots, X_n\}$, timestamp of all samples $T = \{T_1, T_2, T_3, \dots, T_n\}$, window size = w seconds, step size = s seconds
2: Output: Different overlapped window-based snippets $Y = \{Y_1, Y_2, Y_3, \dots, Y_m\}$
3: $k = 1$
4: $m = \lfloor (T_n - T_1) / s \rfloor - w + 1$
5: for $j = 1$ to m do
6:   find $T_p$ in T such that $T_p - T_k \le w \bigwedge T_{p+1} - T_k \ge w$
7:   $Y_j = \{X_k, X_{k+1}, X_{k+2}, \dots, X_{k+p-1}\}$
8:   find $T_r$ in T such that $T_r - T_k \le s \bigwedge T_{r+1} - T_k \ge s$
9:   $k = k + r$
10: end for
```

Datasets with two different numbers of features were created, in order to compare the performance between applying feature extraction and without feature extraction. For Groups 1 to 3, each individual sample consists of nine network features extracted from a GOOSE packet, 18 physical features from various sensor data when the packet is received, and the label based on labelling method type 3. For Groups 4 to 6, each individual sample consists of six network features, seven summarised physical features, and the label.

Meanwhile, to evaluate the performance of applying quantity-based sliding window algorithms, applying time-based sliding window algorithms, and without sliding window algorithms, we divided datasets into an additional three groups. Groups 1 and 4 did not apply sliding window algorithms. Groups 2 and 5 applied time-based sliding window algorithms. Groups 3 and 6 applied quantity-based sliding window algorithms.

All groups of datasets have 10 labels. Label 0 indicates everything is under normal operation. Label 1 means there is a non-malicious event that happens under the emergency operation. Label 901 to label 908 represent various insider attack scenarios described in Table 1. By doing this, our detection model can distinguish different types of attacks as well as two benign behaviours.

**TABLE 3.** Different groups of datasets.

| | Applied sliding window | Labelling methods | # of training scenarios | # of testing scenarios | # of training samples | # of testing samples | # of features | # of different labels |
|---|---|---|---|---|---|---|---|---|
| **Group 1** | No | Type 3 | 23 | 8 | 27919 | 9902 | 27 | 10 |
| **Group 2** | Time-based | Sequential labelling | 23 | 8 | various | various | 27 | 10 |
| **Group 3** | Quantity-based | Sequential labelling | 23 | 8 | various | various | 27 | 10 |
| **Group 4** | No | Type 3 | 23 | 8 | 27919 | 9902 | 13 | 10 |
| **Group 5** | Time-based | Sequential labelling | 23 | 8 | various | various | 13 | 10 |
| **Group 6** | Quantity-based | Sequential labelling | 23 | 8 | various | various | 13 | 10 |

**TABLE 4.** Different machine learning models.

| | Classification Types | Main algorithms | Applied datasets | Hyperparameters |
|---|---|---|---|---|
| **Model 1** | Traditional individual sample classification | SVM | Group 1 and 4 (Shuffled) | Based on MATLAB's library "fitcecoc()", Classifier 'Coding' = 'onevsone' |
| **Model 2** | | KNN | Group 1 and 4 (Shuffled) | Based on MATLAB's library "fitcknn()", Find optimal Numer_Neighbours from range 1 to 10 |
| **Model 3** | | Decision Tree | Group 1 and 4 (Shuffled) | Based on MATLAB's library "fitctree()" |
| **Model 4** | Sequential classification | BiLSTM without sliding window | Group 1 and 4 (Without shuffle) | Layers = [sequenceInputLayer(), bilstmLayer(), fullyConnectedLayer(), softmaxLayer, classificationLayer] Number_HiddenUnits = 200; Max_Epochs = 50 Solver = adam; MiniBatchSize = 256; |
| **Model 5** | | BiLSTM with sliding window | Group 2, 3, 5, 6 (Shuffled) | |

Furthermore, we also implemented a shuffle process to generate holdout samples to avoid overfitting problems in machine learning. For dataset Groups 1 and 4, all individual samples from all scenarios in both training datasets and testing datasets were merged into two separate data sheets respectively and shuffled into random orders. For dataset Groups 2, 3, 5 and 6, all individual samples from each scenario were divided into window-based snippets first, then all snippets from all scenarios were merged into one data sheet and shuffled with random orders.

## VI. MACHINE LEARNING PROCESS

At this stage, we created a total of five machine learning models for evaluation. These models contain three traditional machine learning models and two sequential classification machine learning models which are listed in Table 4. On the one hand, traditional machine learning models usually classify each individual sample with a predefined label. We selected the three most popular traditional machine learning models with high detection performance, and these include support vector machine (SVM), K-nearest neighbour (KNN), and decision tree. On the other hand, sequential classification machine learning models identify a sequence of samples with one particular label. Most sequential classification machine learning models are generally based on recurrent neural networks (RNN), and we chose an improved RNN – bidirectional long short-term memory (BiLSTM). In comparison to the performance of using sliding window algorithms and without sliding window algorithms, we created two corresponding models. All models were created based on MATLAB's existing libraries. The settings of critical hyperparameters from these libraries are shown in Table 4. Additionally, for model 5, we implemented a cross-validation mechanism during the training process to improve the training performance. We divided the original training datasets into 80% training datasets and 20% validation datasets. Also, we set up the validation patience to 5 epochs to avoid overfitting. The training process will stop if the validation loss does not improve for 5 epochs. A detailed analysis of the quality of the learning process in BiLSTM is presented in Section IX.

## VII. EVALUATION RESULTS

We conducted a total of five experiments to train and test different groups of datasets using various machine learning models. In each experiment, the main objective was to train all benign behaviours and stealthy attack behaviours only triggered from IED1, and then to detect the same insider attack behaviours generating from IED2. IED1 and IED2 are the same type of protection relays but protect different sections of the primary plant. As shown in Figure 1, IED1 monitors the status of the three-phase transformer1, detects any abnormal current readings and provides over-current protection to transformer1. Similarly, IED2 protects transformer2. However, if these two devices are compromised during

**TABLE 5.** Different experiments and their objectives.

| Experiment ID | Applied datasets | Applied models | Objective |
|---|---|---|---|
| **Experiment A** | Group 1 | Model 1, 2, 3 | Evaluating the detection performance when applying traditional models without the feature extraction method in this paper |
| **Experiment B** | Group 2<br>Group 3 | Model 5 | Evaluating the detection performance when applying BiLSTM and two sliding window algorithms without feature extraction |
| **Experiment C** | Group 4 | Model 1, 2, 3 | Evaluating the detection performance when applying traditional models with feature extraction |
| **Experiment D** | Group 4 | Model 4 | Evaluating the detection performance when applying BiLSTM and feature extraction, but without sliding window |
| **Experiment E** | Group 5<br>Group 6 | Model 5 | Evaluating the detection performance when applying BiLSTM, two sliding window algorithms, and feature extraction |

**TABLE 6.** The evaluation results of experiment A.

| | Training Accuracy | Testing Accuracy | FP | FN | Others | TP | TN | FPR | FNR | |
|---|---|---|---|---|---|---|---|---|---|---|
| **KNN** | 99.706% | 82.882% | 232 | 815 | 648 | 50 | 8157 | 2.766% | 94.220% | Round1 |
| | 99.703% | 82.842% | 236 | 815 | 648 | 50 | 8153 | 2.813% | 94.220% | Round2 |
| | 99.692% | 82.822% | 238 | 815 | 648 | 50 | 8151 | 2.837% | 94.220% | Round3 |
| | **99.700%** | **82.849%** | **235** | **815** | **648** | **50** | **8154** | **2.801%** | **94.220%** | **Average** |
| **SVM 1V1** | 94.043% | 75.116% | 143 | 494 | 1827 | 463 | 6975 | 2.009% | 51.620% | Round1 |
| | 94.043% | 75.116% | 143 | 513 | 1808 | 463 | 6975 | 2.009% | 52.561% | Round2 |
| | 94.043% | 75.116% | 143 | 493 | 1828 | 463 | 6975 | 2.009% | 51.569% | Round3 |
| | **94.043%** | **75.116%** | **143** | **500** | **1821** | **463** | **6975** | **2.009%** | **51.921%** | **Average** |
| **Decision Tree** | 99.563% | 85.629% | 97 | 705 | 621 | 201 | 8278 | 1.158% | 77.815% | Round1 |
| | 99.556% | 85.629% | 97 | 705 | 621 | 201 | 8278 | 1.158% | 77.815% | Round2 |
| | 99.552% | 85.629% | 97 | 705 | 621 | 201 | 8278 | 1.158% | 77.815% | Round3 |
| | **99.557%** | **85.629%** | **97** | **705** | **621** | **201** | **8278** | **1.158%** | **77.815%** | **Average** |

manufacture or after deployment via a software update, they could be used to launch "insider" attacks within the substation's own infrastructure. Furthermore, the design objective of each experiment is shown in Table 5. The detailed results are discussed below.

## *A. TRADITIONAL MODELS WITHOUT FEATURE EXTRACTION*

In this experiment, we applied three traditional machine learning models – KNN, SVM, and decision tree – to classify different individual samples. For each model we repeated training and testing processes three times and calculated the average performance which is shown in Table 6.

FP is the number of false-positive errors when non-malicious samples are misclassified as malicious ones. FN is the number of false-negative errors when malicious samples are misclassified as non-malicious ones. TP is the number of true positives when malicious samples are identified successfully. TN is the number of true negatives when non-malicious samples are classified successfully. "Others" is the number of non-critical errors when either one benign sample is misclassified as another benign type (e.g., label 0 is misclassified as 1), or one malicious sample is misclassified as another malicious type (e.g., label 908 is misclassified as 907). The False-positive rate (FPR), also called fall-out, indicates how many FPs there are among all non-malicious samples. The False-negative rate (FNR), also called the miss rate, shows how many FNs are among all malicious samples.

From Table 6, without the feature extraction method, all three traditional models show a very high FNR from 51.921% to 94.22% such that most stealthy attacks cannot be detected. This proves that existing methods cannot effectively detect insider attack scenarios triggering from all other IEDs when only learning attacks from one typical IED.

## *B. BiLSTM AND TWO SLIDING WINDOW ALGORITHMS WITHOUT FEATURE EXTRACTION*

In this experiment, we still used the original 27 feature datasets without feature extraction, and applied the sequential classification model – BiLSTM and two sliding window algorithms – to classify a sequence of samples. The training and testing process was repeated four times with different settings of window size and step size.

For time-based sliding window algorithms, the window size and step size are defined as a certain number of seconds. Since GOOSE packets are sent periodically over the process bus, the heartbeats of GOOSE messages are normally one second during normal operation. This means in a cycle of one second, all IEDs publish their GOOSE messages at least once. Thus, the window size and step size are preferred to be an integer value of seconds, so every window-based snippet contains a full cycle of information of all GOOSE packets. The evaluation results are shown in Table 7. From Table 7, the FNRs are from 55% to 67.762%. Thus, without feature extraction, the BiLSTM model with time-based sliding window algorithms is better than traditional machine learning models.

For quantity-based sliding window algorithms, the window size and step size were defined as a certain number of packets. The evaluation results are shown in Table 8. From Table 8, the FNRs are from 37.402% to 55.340%. These results also prove that the BiLSTM model with quantity-based sliding window algorithms is also better than traditional machine learning models.

Additionally, from Table 7 and Table 8, the FNRs when applying quantity-based sliding window algorithms are from 37.402% to 55.340%. These are generally lower than the FNRs with a range of 55% to 67.762% when applying a time-based algorithm. Therefore, we conclude that the quantity-based sliding window algorithms perform better than the time-based ones in the application of detecting stealthy attacks within SASs. Since a 37.402% FNR is not acceptable, only applying BiLSTM and sliding window algorithms are not enough. Thus, additional feature extraction methods are required.

### C. TRADITIONAL MODELS WITH FEATURE EXTRACTION

In this experiment, we still applied three traditional machine learning models to classify different individual samples. However, we also applied a feature extraction method in which a total of 13 features were extracted from the original 27 features. Similar to experiment A, we also repeated training and testing processes three times for each model. The evaluation results are shown in Table 9.

From Table 9, all three traditional models still produce high FNRs. However, compared to the results in experiment A, after applying the feature extraction method, the FNRs of all three models dropped sharply. For instance, the FNR in the KNN model reduced from 94.22% to 31.748% while the FNR in the decision tree model decreased from 77.815% to 30.261%. Therefore, it is believed that the feature extraction method helps generalise critical features, and thus improves the accuracy of detecting stealthy attacks among multiple devices when only attack datasets from one typical device are trained.

Furthermore, the FNRs in experiment C are from 30.261% to 39.619%, which are better than the FNRs in experiment B from 37.402% to 55.340%. From these results, it is inferred that when detecting stealthy insider attacks with a limited volume of datasets, applying the feature extraction method is more important than improving the machine learning model.

### D. BiLSTM AND FEATURE EXTRACTION WITHOUT SLIDING WINDOW

In this experiment, we only applied feature extraction and BiLSTM. The whole sequence of the dataset was trained using BiLSTM without dividing it into different window-based snippets. Since time-serial patterns of datasets are important during the training process, the datasets were not shuffled. Different from applying sliding window algorithms, the machine learning model still classifies each sample with a particular label. The training and testing process was repeated three times.

The evaluation results are shown in Table 10. From Table 10, the FNR is always 100%. Therefore, it is obvious that without sliding window algorithms, the BiLSTM model even with feature extraction cannot detect any stealthy attack scenarios. The reason may be that almost 97% of samples in the training datasets were labelled as benign behaviours, and the model ignores those 3% samples of malicious behaviours during the learning process. Due to the special characteristic of SASs that every IEDs publishes GOOSE messages repeatedly, unless all IEDs are compromised, the system behaviours of SASs are usually unbalanced so that most samples are benign. Therefore, regarding the unbalanced network traffic when FDIAs occur within SASs, it is essential to apply sliding window algorithms for sequential classification to detect stealthy attacks accurately.

### E. BiLSTM AND TWO SLIDING WINDOW ALGORITHMS WITH FEATURE EXTRACTION

Finally, we applied feature extraction, BiLSTM, and two sliding window algorithms to classify a sequence of samples. The training and testing process was repeated four times with different settings of window size and step size.

Table 11 demonstrates the evaluation results of applying time-based sliding window algorithms. From Table 11, the FNR is almost reduced to 22.984% when the window size is three seconds, and the step size is one second. This FNR is better than the 55% FNR when applying time-based sliding window algorithms in experiment B, and even better than the 37.402% FNR when applying quantity-based sliding window algorithms in experiment B. This result proves the importance of applying feature extraction with BiLSTM and sliding window algorithms to detect stealthy attacks when only learning attack datasets from one typical device.

Table 12 displays the evaluation results of applying quantity-based sliding window algorithms. From Table 12, when the window size is 12 packets, and the step size is one packet, the FNR is reduced to 5.385%. This is the best result among all the experiments. This result again proves that

**TABLE 7.** The evaluation results of experiment B when applying time-based sliding windows.

| | Training Accuracy | Validation Accuracy | Testing Accuracy | FP | FN | Others | TP | TN | FPR | FNR | Window size (seconds) | Step size (seconds) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM with Time-based Sliding Window | 97.660% | 98.020% | 81.850% | 0 | 232 | 183 | 155 | 1717 | 0.000% | 59.948% | 2 | 1 |
| | 96.880% | 98.530% | 80.340% | 0 | 198 | 250 | 162 | 1689 | 0.000% | 55.000% | 3 | 1 |
| | 96.090% | 96.890% | 79.070% | 0 | 330 | 122 | 157 | 1661 | 0.000% | 67.762% | 4 | 1 |
| | 97.660% | 97.100% | 77.850% | 0 | 262 | 239 | 137 | 1624 | 0.000% | 65.664% | 5 | 1 |

**TABLE 8.** The evaluation results of experiment B when applying quantity-based sliding windows.

| | Training Accuracy | Validation Accuracy | Testing Accuracy | FP | FN | Others | TP | TN | FPR | FNR | Window size (packets) | Step size (packets) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM with Quantity-based Sliding window | 96.880% | 98.230% | 79.830% | 1 | 890 | 1095 | 1049 | 6811 | 0.015% | 45.900% | 8 | 1 |
| | 99.220% | 97.190% | 78.630% | 1 | 285 | 241 | 230 | 1709 | 0.058% | 55.340% | 8 | 4 |
| | 98.830% | 98.140% | 81.130% | 9 | 766 | 1077 | 1282 | 6680 | 0.135% | 37.402% | 12 | 1 |
| | 97.060% | 98.050% | 78.970% | 2 | 282 | 233 | 266 | 1675 | 0.119% | 51.460% | 12 | 4 |

**TABLE 9.** The evaluation results of experiment C.

| | Training Accuracy | Testing Accuracy | FP | FN | Others | TP | TN | FPR | FNR | |
|---|---|---|---|---|---|---|---|---|---|---|
| KNN | 99.262% | 90.325% | 217 | 247 | 494 | 531 | 8413 | 2.514% | 31.748% | Round1 |
| | 99.327% | 90.436% | 212 | 245 | 490 | 533 | 8422 | 2.455% | 31.491% | Round2 |
| | 99.366% | 90.497% | 206 | 248 | 487 | 530 | 8431 | 2.385% | 31.877% | Round3 |
| | **99.318%** | **90.419%** | **212** | **247** | **490** | **531** | **8422** | **2.455%** | **31.748%** | **Average** |
| SVM 1V1 | 84.426% | 81.095% | 145 | 396 | 1331 | 602 | 7428 | 1.915% | 39.679% | Round1 |
| | 84.426% | 81.125% | 145 | 393 | 1331 | 605 | 7428 | 1.915% | 39.379% | Round2 |
| | 84.426% | 81.064% | 145 | 397 | 1333 | 599 | 7428 | 1.915% | 39.859% | Round3 |
| | **84.426%** | **81.095%** | **145** | **395** | **1332** | **602** | **7428** | **1.915%** | **39.619%** | **Average** |
| Decision Tree | 99.244% | 92.830% | 208 | 302 | 200 | 696 | 8496 | 2.390% | 30.261% | Round1 |
| | 99.262% | 92.860% | 207 | 302 | 198 | 696 | 8499 | 2.378% | 30.261% | Round2 |
| | 99.237% | 92.850% | 208 | 302 | 198 | 696 | 8498 | 2.389% | 30.261% | Round3 |
| | **99.248%** | **92.847%** | **208** | **302** | **199** | **696** | **8498** | **2.389%** | **30.261%** | **Average** |

**TABLE 10.** The evaluation results of experiment D.

| | Training Accuracy | Testing Accuracy | FP | FN | Others | TP | TN | FPR | FNR | |
|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM without Sliding Window | 93.480% | 89.900% | 0 | 1000 | 0 | 0 | 8902 | 0.000% | 100.000% | Round1 |
| | 94.850% | 89.880% | 0 | 1000 | 2 | 0 | 8754 | 0.000% | 100.000% | Round2 |
| | 93.730% | 89.590% | 0 | 1000 | 31 | 0 | 8871 | 0.000% | 100.000% | Round3 |
| | **94.020%** | **89.790%** | **0** | **1000** | **11** | **0** | **8842** | **0.000%** | **100.000%** | **Average** |

**TABLE 11.** The evaluation results of experiment E when applying time-based sliding windows.

| | Training Accuracy | Validation Accuracy | Testing Accuracy | FP | FN | Others | TP | TN | FPR | FNR | Window size (seconds) | Step size (seconds) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM with Time-based Sliding Window | 98.440% | 96.910% | 87.190% | 45 | 161 | 87 | 320 | 1674 | 2.618% | 33.472% | 2 | 1 |
| | 97.660% | 96.680% | 90.260% | 0 | 114 | 108 | 382 | 1675 | 0.000% | 22.984% | 3 | 1 |
| | 96.480% | 96.370% | 87.530% | 0 | 130 | 153 | 346 | 1641 | 0.000% | 27.311% | 4 | 1 |
| | 97.660% | 97.770% | 91.470% | 0 | 132 | 61 | 401 | 1668 | 0.000% | 24.765% | 5 | 1 |

the quantity-based sliding window algorithms perform better than the time-based ones when detecting stealthy attacks in SASs.

Furthermore, in Table 8 and Table 12, when applying BiLSTM with quantity-based sliding window algorithms in two different experiments, the smallest FNR is always shown

**TABLE 12.** The evaluation results of experiment E when applying quantity-based sliding windows.

| | Training Accuracy | Validation Accuracy | Testing Accuracy | FP | FN | Others | TP | TN | FPR | FNR | Window size (packets) | Step size (packets) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM with Quantity-based Sliding window | 98.050% | 97.890% | 93.630% | 4 | 197 | 426 | 2179 | 7040 | 0.057% | 8.291% | 8 | 1 |
| | 96.480% | 96.470% | 88.360% | 1 | 183 | 103 | 419 | 1760 | 0.057% | 30.399% | 8 | 4 |
| | 98.830% | 97.830% | 94.220% | 3 | 133 | 431 | 2337 | 6910 | 0.043% | 5.385% | 12 | 1 |
| | 98.050% | 95.880% | 88.570% | 2 | 128 | 151 | 493 | 1684 | 0.119% | 20.612% | 12 | 4 |

under the same configuration – when the window size is 12 packets and the step size is one packet. Therefore, different configurations of window size and step size influence the detection accuracy. Additionally, from the results, it is assumed that larger window size and smaller step size may present better performance. Thus, a comprehensive analysis of the performance of various window sizes and step sizes was conducted and is discussed in the next section.

In conclusion, these experiments testify that applying feature extraction, BiLSTM, and quantity-based sliding window algorithms can effectively detect stealthy attacks triggering from similar untrusted IEDs when only learning malicious behaviours from one typical IED in the process bus. Although the 5.385% FNR is still high for anomaly detection in critical infrastructure, it is greatly improved from 51.921% in experiment A, when only traditional machine learning models were applied without feature extraction.

## VIII. RECOMMENDED WINDOW SIZE AND STEP SIZE

In this section, we conducted two additional experiments to determine the recommended setting of window size and step size in sliding window algorithms. Since previous experimental results show that quantity-based algorithms are better than time-based algorithms, we only focused on the settings in quantity-based sliding window algorithms.

### A. FIXED WINDOW SIZE AND VARIOUS STEP SIZE

For experiment F, the step size was researched. We applied BiLSTM with quantity-based sliding windows to train and test dataset Group 6. With a fixed window size of 24 packets and various step sizes, we observed how different step sizes influence the detection performance. The number of testing samples and the average detection time per sample were recorded. We collected the total testing time in milliseconds, then divided it by the number of testing samples to get the average detection time per sample. Table 13 shows the performance of various configurations.

From Table 13, when the step size increases from 1 to 12, consistently, the number of training samples reduces from 9718 to 813, the FNR increases from 1.666% to 43.304%, and the average detection time per sample increases from 2.6358 to 9.4726 milliseconds. Similarly, the same pattern happens when the window size is 16. Therefore, regarding the FNR and average detection time, the preferred configuration of step size is the smallest, i.e., one packet.

The reason is related to the number of training samples. From Table 13, when the step size is doubled, the number of testing samples is almost reduced to half. Accordingly, the number of training samples is also reduced to half. Without a sufficient number of training samples, the training process will not perform well, and thus, leads to high FNR.

### B. FIXED STEP SIZE AND VARIOUS WINDOW SIZE

For experiment G, the window size was researched. With a fixed step size and various window sizes, we observed how different window sizes influence the detection performance. Based on the assumption of the previous experiment, the step size was set up as one packet to get the lowest FNR. Table 14 shows the performance of various configurations.

From Table 14, when the window size increases from 8 to 30, the number of testing samples reduces slightly, and the average detection time per sample increases. However, the FNRs are varied without an obvious pattern due to the uncertainty and randomisation of the learning process. We illustrated different FNRs regarding different window sizes in Figure 4. From Figure 4, generally, the FNR is improved when the window size increases. When the step size is from 20 to 30, all the FNRs are below 4% which is assumed to be the preferred configuration range. Most importantly, when the step size is 22, the FNR is 0.37%, which is the lowest among all the experiments.

Furthermore, we also illustrated different detection times regarding different window sizes in Figure 5. From Figure 5, the detection time keeps increasing when the window size increases. Since IEC 61850 compliant SASs have a strict requirement for the response time of 3 milliseconds, the detection time should also be less than 3 milliseconds. According to this requirement, when the window size is larger than 28, the detection time is larger than 3 milliseconds which is not acceptable for anomaly detection within SASs. Therefore, the window size is suggested to be less than 28.

As a result, in our simulation environment, considering both the detection time and detection accuracy, the recommended window size is from 20 to 28 packets while the preferred step size is one packet.

## IX. DISCUSSION

Based on our experimental results, we can conclude that by applying feature extraction, BiLSTM, and quantity-based sliding window algorithms, our approach can effectively detect stealthy attacks triggering from similar untrusted IEDs

TABLE 13. The evaluation results of experiment F – various step sizes with a fixed window size.

| Training Accuracy | Validation Accuracy | Testing Accuracy | # of testing samples | FP | FN | Others | TP | TN | FPR | FNR | Detection time (ms) per sample | Window size | Step size |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97.27% | 98.27% | 95.54% | 9718 | 3 | 47 | 383 | 2774 | 6511 | 0.046% | 1.666% | 2.6358 | 24 | 1 |
| 98.44% | 97.96% | 92.00% | 4863 | 2 | 131 | 246 | 1228 | 3256 | 0.061% | 9.639% | 2.9818 | 24 | 2 |
| 97.66% | 98.03% | 92.78% | 3242 | 1 | 98 | 135 | 831 | 2177 | 0.046% | 10.549% | 3.4739 | 24 | 3 |
| 98.05% | 97.89% | 90.35% | 2434 | 1 | 119 | 115 | 569 | 1630 | 0.061% | 17.297% | 4.0734 | 24 | 4 |
| 97.66% | 96.72% | 90.20% | 1623 | 0 | 94 | 65 | 368 | 1096 | 0.000% | 20.346% | 5.5086 | 24 | 6 |
| 95.70% | 95.78% | 84.50% | 1219 | 1 | 113 | 75 | 234 | 796 | 0.125% | 32.565% | 6.7741 | 24 | 8 |
| 96.09% | 93.90% | 81.30% | 813 | 1 | 97 | 54 | 127 | 534 | 0.187% | 43.304% | 9.4726 | 24 | 12 |

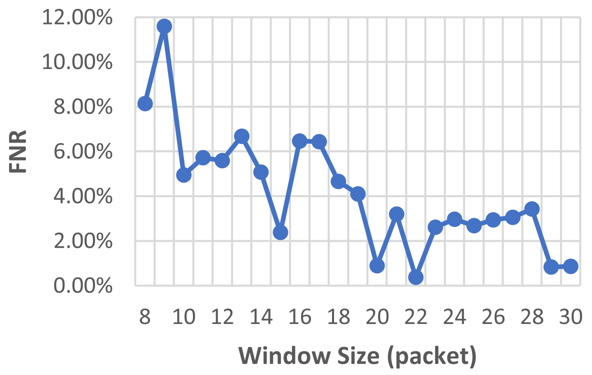

FIGURE 4. Detection FNRs when selecting different window sizes.

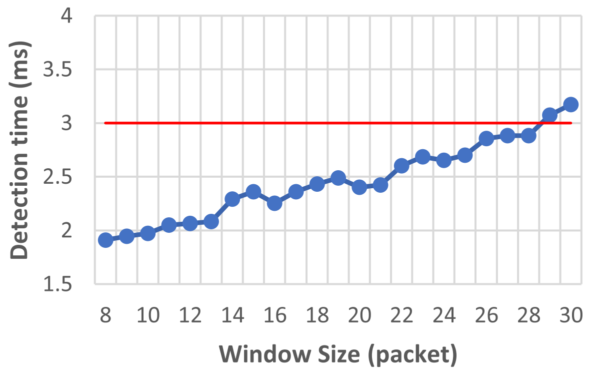

FIGURE 5. Detection times when selecting different window sizes.

when only learning malicious behaviours from one IED on the process bus. When the window size is 22 packets and the step size is 1 packet, we achieved the best results in which the FNR reduced to 0.372%. Furthermore, we conducted a further assessment of our anomaly detection methods, such as the time efficiency of algorithms, unbalanced dataset issue, and the quality of the learning process.

### A. TIME EFFICIENCY OF ALGORITHMS

Similar to other researchers' findings, our experimental results also indicate that there is a trade-off between detection accuracy and detection time when selecting different window sizes. When the window size increases, the waiting time for obtaining all samples of one window-based sequence increases, thus the detection time increases accordingly. Figure 5 demonstrates this pattern. Due to the strict requirement of 3 millisecond response time in SASs, the window size should be limited to satisfy this requirement even though a larger window size gives more accuracy. Additionally, we trained and tested all datasets offline. After observing the total testing time, we divided it by the total number of testing samples to get the average testing time per sample. This average testing time reflects the average detection time when the detection model is online. However, all training and testing processes were run in a simulation environment with a single CPU. If the detection system is deployed in a dedicated computer, it is believed that the detection time could be less than our current results.

### B. UNBALANCED DATASETS

According to the results in experiment D, it is important to apply sliding window algorithms for sequential classification algorithms, especially when datasets are unbalanced between benign behaviours and malicious behaviours. Generally, due to the specific behaviours of the IEC 61850 compliant substation that various IEDs publish GOOSE messages to the process bus, the ratio of benign packets and malicious packets is usually unbalanced. Unless all IEDs have been compromised, the number of benign packets is much more than malicious ones. Therefore, the detection model needs to learn unbalanced datasets to detect anomalies in real systems' environments.

### C. BiLSTM OR LSTM

Compared to the LSTM model which only learns datasets from the forward direction, the BiLSTM model learns datasets from both forward and backward directions. Therefore, BiLSTM usually has more complex neural networks than LSTM, and accordingly, requires more time and resources for the training and testing process. However, it is important to apply BiLSTM to detect stealthy insider attack scenarios within SASs as it gives higher detection accuracy.

The BiLSTM model learns the contexts from two directions. It helps the detection algorithms to understand what

**TABLE 14.** **The evaluation results of experiment G – various window sizes with a fixed step size.**

| Training Accuracy | Validation Accuracy | Testing Accuracy | # of testing samples | FP | FN | Others | TP | TN | FPR | FNR | Detection time (ms) per sample | Window size | Step size |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 97.66% | 97.95% | 93.98% | 9846 | 4 | 196 | 393 | 2212 | 7041 | 0.057% | 8.140% | 1.9104 | 8 | 1 |
| 98.05% | 97.62% | 92.97% | 9838 | 1 | 280 | 411 | 2135 | 7011 | 0.014% | 11.594% | 1.9455 | 9 | 1 |
| 98.44% | 97.74% | 94.83% | 9830 | 1 | 122 | 385 | 2344 | 6978 | 0.014% | 4.947% | 1.9715 | 10 | 1 |
| 98.44% | 98.14% | 93.39% | 9822 | 9 | 136 | 504 | 2239 | 6934 | 0.130% | 5.726% | 2.0495 | 11 | 1 |
| 96.88% | 97.56% | 93.25% | 9814 | 3 | 133 | 526 | 2243 | 6909 | 0.043% | 5.598% | 2.0634 | 12 | 1 |
| 98.05% | 97.78% | 93.12% | 9806 | 7 | 162 | 506 | 2261 | 6870 | 0.102% | 6.686% | 2.0814 | 13 | 1 |
| 96.48% | 98.37% | 94.13% | 9798 | 1 | 127 | 447 | 2377 | 6848 | 0.015% | 5.072% | 2.2903 | 14 | 1 |
| 97.27% | 98.28% | 95.04% | 9790 | 2 | 61 | 423 | 2493 | 6811 | 0.029% | 2.388% | 2.3616 | 15 | 1 |
| 99.22% | 97.72% | 93.67% | 9782 | 2 | 165 | 452 | 2386 | 6777 | 0.030% | 6.468% | 2.2531 | 16 | 1 |
| 98.83% | 98.31% | 94.13% | 9774 | 1 | 169 | 404 | 2458 | 6742 | 0.015% | 6.433% | 2.3593 | 17 | 1 |
| 97.66% | 97.98% | 94.45% | 9766 | 0 | 123 | 419 | 2513 | 6711 | 0.000% | 4.666% | 2.4309 | 18 | 1 |
| 99.22% | 98.24% | 94.08% | 9758 | 0 | 107 | 471 | 2498 | 6682 | 0.000% | 4.107% | 2.4882 | 19 | 1 |
| 99.22% | 98.25% | 95.36% | 9750 | 1 | 24 | 427 | 2650 | 6648 | 0.015% | 0.898% | 2.4031 | 20 | 1 |
| 97.66% | 98.11% | 93.62% | 9742 | 4 | 83 | 535 | 2512 | 6608 | 0.060% | 3.198% | 2.4215 | 21 | 1 |
| 96.88% | 98.09% | 95.02% | 9734 | 11 | 10 | 464 | 2677 | 6572 | 0.167% | 0.372% | 2.6005 | 22 | 1 |
| 98.05% | 97.56% | 93.33% | 9726 | 8 | 68 | 573 | 2535 | 6542 | 0.122% | 2.612% | 2.6847 | 23 | 1 |
| 99.22% | 98.10% | 93.29% | 9718 | 12 | 82 | 439 | 2680 | 6505 | 0.184% | 2.969% | 2.6521 | 24 | 1 |
| 96.48% | 98.03% | 94.72% | 9710 | 11 | 75 | 427 | 2724 | 6473 | 0.170% | 2.680% | 2.7013 | 25 | 1 |
| 98.83% | 97.33% | 92.94% | 9702 | 4 | 78 | 603 | 2570 | 6447 | 0.062% | 2.946% | 2.8542 | 26 | 1 |
| 98.44% | 97.73% | 94.01% | 9694 | 4 | 85 | 492 | 2701 | 6412 | 0.062% | 3.051% | 2.8822 | 27 | 1 |
| 99.22% | 98.13% | 95.24% | 9686 | 11 | 101 | 349 | 2851 | 6374 | 0.172% | 3.421% | 2.8832 | 28 | 1 |
| 98.83% | 98.04% | 94.86% | 9678 | 12 | 24 | 461 | 2841 | 6340 | 0.189% | 0.838% | 3.0728 | 29 | 1 |
| 98.44% | 98.33% | 94.78% | 9670 | 11 | 25 | 469 | 2864 | 6301 | 0.174% | 0.865% | 3.1725 | 30 | 1 |

happens before an anomaly and what happens after an anomaly. Knowing what happens before an anomaly helps predict the following behaviours, while knowing what happens after an anomaly helps validate the current classification, and thus provides more accurate results. In power systems, when a non-malicious fault happens, the instantaneous behaviours might be similar to a stealthy attack scenario that mimics when a non-malicious fault occurs. However, their following behaviours may be different. Thus, with backward learning, the BiLSTM can identify such stealthy attacks within SASs accurately.

Furthermore, in experiment H, we repeated experiment G and only changed the BiLSTM model to the LSTM model. The evaluation results are shown in Table 15. From Table 15, the FNRs are from 13.331% to 27.338%, which are all higher than the worst result 11.594% in Table 14. Therefore, it is obvious that BiLSTM performs better than LSTM when detecting stealthy insider attack scenarios within SASs.

### *D. TIME-BASED OR QUANTITY-BASED*

Based on the results from experiments B and E, it is obvious that the quantity-based sliding window algorithms perform better than the time-based ones in the application of detecting insider attacks within SASs. There are three reasons why we selected quantity-based sliding window algorithms.

Firstly, according to our findings, the smallest step size produces the best accuracy with the lowest FNR. When the time-based sliding window is applied, the smallest step size is one second which involves at least n packets where n is the number of IEDs in the process bus. On the other hand, the smallest step size for the quantity-based sliding window is one packet which is obviously smaller than n packets. Thus, when both algorithms choose the smallest step size, quantity-based shows lower FNR than time-based.

Secondly, according to the complexity of two sliding window algorithms, the time-based approach needs to find the edges of each window by calculating the packet arrival time, and thus require more time to generate window-based datasets than quantity-based.

Thirdly, during training, the sequential classification model usually groups the training data into mini-batches and pads the sequences to ensure they have the same length. For quantity-based sliding window algorithms, every window has the same number of samples, thus it does not require

**TABLE 15.** The evaluation results of experiment H – using lstm instead of BiLSTM.

| Training Accuracy | Validation Accuracy | Testing Accuracy | # of testing samples | FP | FN | Others | TP | TN | FPR | FNR | Detection time (ms) per sample | Window size | Step size |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 96.88% | 96.58% | 87.40% | 9846 | 4 | 608 | 629 | 1616 | 6989 | 0.057% | 27.338% | 2.122689 | 8 | 1 |
| 98.44% | 96.07% | 88.05% | 9838 | 2 | 563 | 611 | 1864 | 6798 | 0.029% | 23.197% | 2.215898 | 9 | 1 |
| 96.88% | 97.13% | 90.67% | 9830 | 5 | 504 | 408 | 1939 | 6974 | 0.072% | 20.630% | 2.292981 | 10 | 1 |
| 95.70% | 96.57% | 88.99% | 9822 | 11 | 521 | 549 | 1813 | 6928 | 0.159% | 22.322% | 2.430259 | 11 | 1 |
| 96.88% | 96.89% | 89.46% | 9814 | 11 | 395 | 628 | 1986 | 6794 | 0.162% | 16.590% | 2.276340 | 12 | 1 |
| 97.27% | 97.45% | 91.32% | 9806 | 9 | 424 | 418 | 2087 | 6868 | 0.131% | 16.886% | 2.581073 | 13 | 1 |
| 96.88% | 97.68% | 89.98% | 9798 | 10 | 309 | 663 | 1979 | 6837 | 0.146% | 13.505% | 2.707695 | 14 | 1 |
| 96.88% | 97.90% | 91.63% | 9790 | 2 | 341 | 476 | 2217 | 6754 | 0.030% | 13.331% | 2.844740 | 15 | 1 |
| 96.88% | 97.68% | 89.44% | 9782 | 11 | 315 | 707 | 2020 | 6729 | 0.163% | 13.490% | 2.918626 | 16 | 1 |
| 97.66% | 96.53% | 86.81% | 9774 | 6 | 706 | 577 | 1918 | 6567 | 0.091% | 26.905% | 2.78187 | 17 | 1 |
| 98.05% | 96.88% | 86.91% | 9766 | 10 | 584 | 684 | 1967 | 6521 | 0.153% | 22.893% | 3.113864 | 18 | 1 |
| 98.05% | 97.53% | 88.60% | 9758 | 10 | 580 | 522 | 1975 | 6671 | 0.150% | 22.701% | 3.241443 | 19 | 1 |
| 98.05% | 97.18% | 87.90% | 9750 | 0 | 554 | 626 | 2097 | 6473 | 0.000% | 20.898% | 3.383590 | 20 | 1 |

**TABLE 16.** The evaluation results of experiment I – different numbers of hidden units.

| Training Accuracy | Validation Accuracy | Testing Accuracy | # of testing samples | FP | FN | Others | TP | TN | FPR | FNR | Detection time (ms) per sample | # of Hidden Units |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 96.88% | 97.20% | 91.81% | 9830 | 6 | 245 | 554 | 2060 | 6965 | 0.086% | 10.629% | 1.466693 | 50 |
| 97.73% | 97.89% | 93.96% | 9830 | 4 | 186 | 404 | 2261 | 6975 | 0.057% | 7.601% | 1.699380 | 100 |
| 99.22% | 97.76% | 92.13% | 9830 | 2 | 214 | 558 | 2079 | 6977 | 0.029% | 9.333% | 1.706255 | 150 |
| 99.61% | 99.33% | 96.33% | 9830 | 9 | 172 | 180 | 2500 | 6969 | 0.129% | 6.437% | 1.809660 | 200 |
| 98.83% | 97.87% | 94.04% | 9830 | 4 | 166 | 416 | 2276 | 6968 | 0.057% | 6.798% | 1.952797 | 250 |
| 96.88% | 98.27% | 93.61% | 9830 | 12 | 158 | 458 | 2235 | 6967 | 0.172% | 6.603% | 2.144731 | 300 |
| 98.05% | 98.21% | 94.21% | 9830 | 3 | 162 | 394 | 2295 | 6976 | 0.043% | 6.593% | 2.270719 | 350 |
| 97.66% | 97.96% | 93.96% | 9830 | 4 | 168 | 402 | 2282 | 6974 | 0.057% | 6.857% | 2.331642 | 400 |

additional padding. On the contrary, for time-based sliding window algorithms, each window has the same time interval, but different amounts of samples, thus requiring additional padding with more computational costs.

Due to these three reasons, quantity-based sliding window algorithms are better than time-based when applying sequential classification algorithms to detect anomalies within SASs.

## *E. THE QUALITY OF LEARNING PROCESS*

Overfitting is a common issue when applying machine learning. The detection results are too close to a particular dataset, and may fail to fit unseen datasets. In this paper, we applied three techniques to mitigate the risk of overfitting.

Firstly, we applied cross-validation during the learning process, and the validation datasets were randomly generated from the whole training datasets. The validation datasets were different among each individual training process. Secondly, we shuffled both the training datasets and validation datasets before the training process started. This "holdout" process will disorder the sequence of datasets and avoid overfitting. Thirdly, we set up the validation tolerance to be 5 epochs. This means that the training process will stop if the validation loss does not improve for 5 epochs. This strategy also mitigates the risk of overfitting.

Furthermore, we selected proper hyper-parameters by trial and error. Table 16 shows the performance when choosing different numbers of hidden units in the BiLSTM layer. The window size is 10 packets, and the step size is one packet. From Table 16, when the number of hidden units increases, the FNR decreases, and the detection time per sample increases. However, when the number of hidden units increases from 200 to 400, the FNR did not show obvious improvement. Therefore, considering both the accuracy and the detection time, the preferred number of hidden units is 200. Similarly, we set up the minibatch size to be 256 regarding the trade-off between the accuracy and the detection time.

### F. MITIGATION OF DRAWBACKS WHEN ONLY LEARNING DATASETS FROM A SAMPLING DEVICE

In this paper, we presented a feature extraction method to solve a generalisation issue to support detecting anomalies from all IEDs while only learning datasets from one typical sampling IED. However, after removing some features, we also lose identity information. Thus, the detection model can only indicate there is an anomaly within an SAS, but cannot directly identify which devices triggered the anomaly. This drawback can be mitigated in two different ways.

Firstly, after the initial detection stage when an anomaly is detected, we can implement an additional process to re-extract the identity information from the abnormal packets. Then, we can discover which devices caused the anomalies. Secondly, by extracting the average value of each level in the primary plant, we can observe if any of these levels involves anomalies. Similarly, we can extract additional physical features to indicate which specific parts of the primary plant have been impacted, such as the standard deviation value of the same level. This aspect of the problem will be considered further in future work.

### G. ERROR TYPE "OTHERS"

Lastly, in each experiment, we indicated the number of special classification errors, called "Others". This type of error is a non-critical one that has two cases. For Case 1, a benign sample is misclassified as another benign type, e.g., label 0 (normal operation status) is misclassified as label 1 (emergency operation status). For Case 2, a malicious sample is misclassified as another malicious type, e.g., label 908 is misclassified as 907. Case 1 will impact the systems' normal operation as it will mislead systems' statuses. However, Case 2 will not bring any impacts to the systems' normal operation as anomalies are still detected eventually, though it may interfere with mitigation decisions as anomalies are misclassified.

To investigate the impacts of this "Other" error, we analysed the detailed results of each experiment in Table 14. For all results with different window sizes, we discovered that Case 1 occurs once at most, and all the rest of the errors are Case 2. Therefore, these "Other" classification errors are of little consequence overall. Nonetheless, this type of error will be investigated further in future work.

## X. CONCLUSION

In this paper, we presented an anomaly detection model to detect insider attacks triggered from untrusted control devices within SASs. Our model combined feature selection and extraction methods, sequential classification algorithms, and sliding window algorithms. By selecting and extracting six critical network features and seven summarised physical features, our model can effectively detect insider attacks from any IED even though malicious behaviours from only one typical IED were learnt. Compared to traditional individual sample classification methods, our method combines BiLSTM and quantity-based sliding window algorithms, and improves detection accuracy by reducing the FNR from 30.261% to 0.372%. In future work, we will extend this method to protect SV communication and also the communication in the high-level station bus of IEC 61850-compliant SASs.

## REFERENCES

[1] J. Hong and C.-C. Liu, "Intelligent electronic devices with collaborative intrusion detection systems," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 271–281, Jan. 2019, doi: 10.1109/TSG.2017.2737826.

[2] Y. Kwon, S. Lee, R. King, J. Lim, and H. Kim, "Behavior analysis and anomaly detection for a digital substation on cyber-physical system," *Electronics*, vol. 8, no. 3, p. 326, Mar. 2019, doi: 10.3390/electronics8030326.

[3] Y. Yang, H.-Q. Xu, L. Gao, Y.-B. Yuan, K. McLaughlin, and S. Sezer, "Multidimensional intrusion detection system for IEC 61850-based SCADA networks," *IEEE Trans. Power Del.*, vol. 32, no. 2, pp. 1068–1078, Apr. 2017, doi: 10.1109/TPWRD.2016.2603339.

[4] L. E. da Silva and D. V. Coury, "A new methodology for real-time detection of attacks in IEC 61850-based systems," *Electr. Power Syst. Res.*, vol. 143, pp. 825–833, Feb. 2017, doi: 10.1016/j.epsr.2016.08.022.

[5] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014, doi: 10.1109/SURV.2013.052213.00046.

[6] Z. Ouyang, X. Sun, and D. Yue, "Hierarchical time series feature extraction for power consumption anomaly detection," in *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration*, K. Li, Y. Xue, S. Cui, Q. Niu, Z. Yang, P. Luk, Eds. Singapore: Springer, 2017, pp. 267–275.

[7] A. Valdes, R. Macwan, and M. Backes, "Anomaly detection in electrical substation circuits via unsupervised machine learning," in *Proc. IEEE 17th Int. Conf. Inf. Reuse Integr. (IRI)*, Pittsburgh, PA, USA, May 2016, pp. 500–505, doi: 10.1109/IRI.2016.74.

[8] P. Kreimel, O. Eigner, F. Mercaldo, A. Santone, and P. Tavolato, "Anomaly detection in substation networks," *J. Inf. Secur. Appl.*, vol. 54, Oct. 2020, Art. no. 102527, doi: 10.1016/j.jisa.2020.102527.

[9] D. S. Smys, D. A. Basar, and D. H. Wang, "Hybrid intrusion detection system for Internet of Things (IoT)," *December*, vol. 2, no. 4, pp. 190–199, Sep. 2020, doi: 10.36548/jismac.2020.4.002.

[10] H. Qu, L. Lei, X. Tang, and P. Wang, "A lightweight intrusion detection method based on fuzzy clustering algorithm for wireless sensor networks," *Adv. Fuzzy Syst.*, vol. 2018, pp. 1–12, Jun. 2018, doi: 10.1155/2018/4071851.

[11] R. Leszczyna, "A review of standards with cybersecurity requirements for smart grid," *Comput. Secur.*, vol. 77, pp. 262–276, Aug. 2018, doi: 10.1016/j.cose.2018.03.011.

[12] C.-C. Sun, A. Hahn, and C.-C. Liu, "Cyber security of a power grid: State-of-the-art," *Int. J. Elect. Power Energy Syst.*, vol. 99, pp. 45–56, Jul. 2018, doi: 10.1016/j.ijepes.2017.12.020.

[13] *Communication Networks and Systems for Power Utility Automation*, document 61850, IEC, 2003.

[14] M. Kezunovic, Y. Guan, C. Guo, and M. Ghavami, "The 21st century substation design: Vision of the future," in *Proc. IREP Symp. Bulk Power Syst. Dyn. Control (IREP)*, Rio de Janeiro, Brazil, 2010, pp. 1–8, doi: 10.1109/IREP.2010.5563267.

[15] *Power Systems Management and Associated Information Exchange—Data and Communications Security—Introduction to Security Issues*, document IEC 62351-1:2007, IEC Technical Committee, 2007.

[16] A. Elgargouri and M. Elmusrati, "Analysis of cyber-attacks on IEC 61850 networks," in *Proc. IEEE 11th Int. Conf. Appl. Inf. Commun. Technol.*, Moscow, Russia, Jun. 2017, pp. 1–4, doi: 10.1109/ICAICT.2017.8686894.

[17] R. Grandgenett, R. Gandhi, and W. Mahoney, "Exploitation of Allen Bradley's implementation of EtherNet/IP for denial of service against industrial control systems," in *Proc. 9th Int. Conf. Cyber Warfare Secur.*, West Lafayette, IN, USA, 2014, pp. 58–65.

[18] H. Yoo and T. Shon, "Challenges and research directions for heterogeneous cyber-physical system based on IEC 61850: Vulnerabilities, security requirements, and security architecture," *Future Gener. Comput. Syst.*, vol. 61, pp. 128–136, Oct. 2016, doi: 10.1016/j.future.2015.09.026.

[19] S. Samtani, S. Yu, H. Zhu, M. Patton, and H. Chen, ''Identifying SCADA vulnerabilities using passive and active vulnerability assessment techniques,'' in *Proc. IEEE Conf. Intell. Secur. Inform. (ISI)*, Tucson, AZ, USA, May 2016, pp. 25–30, doi: 10.1109/ISI.2016.7745438.

[20] R. Kalayappan and S. R. Sarangi, ''SecCheck: A trustworthy system with untrusted components,'' in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Pittsburgh, PA, USA, Dec. 2016, pp. 379–384, doi: 10.1109/ISVLSI.2016.31.

[21] S. Sethumadhavan, A. Waksman, M. Suozzo, Y. Huang, and J. Eum, ''Trustworthy hardware from untrusted components,'' *Commun. ACM*, vol. 58, no. 9, pp. 60–71, Aug. 2015, doi: 10.1145/2699412.

[22] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer, ''Threat analysis of BlackEnergy malware for synchrophasor based real-time control and monitoring in smart grid,'' in *Proc. 4th Int. Symp. ICS SCADA Cyber Secur. Res.*, Belfast, U.K., Aug. 2016, pp. 53–63, doi: 10.14236/ewic/ICS2016.7.

[23] *Asset Management—Overview, Principles and Terminology*, document ISO 55000 ISO/TC 251, 2014.

[24] F. Xie, Y. Peng, W. Zhao, Y. Gao, and X. Han, ''Evaluating industrial control devices security: Standards, technologies and challenges,'' in *Computer Information Systems and Industrial Management*, K. Saeed and V. Snéel, Eds. Berlin, Germany: Springer, 2014, pp. 624–635.

[25] Y. M. Khaw, A. Abiri Jahromi, M. F. M. Arani, S. Sanner, D. Kundur, and M. Kassouf, ''A deep learning-based cyberattack detection system for transmission protective relays,'' *IEEE Trans. Smart Grid*, vol. 12, no. 3, pp. 2554–2565, May 2021, doi: 10.1109/TSG.2020.3040361.

[26] Q. Huang, S. Jing, J. Li, D. Cai, J. Wu, and W. Zhen, ''Smart substation: State of the art and future development,'' *IEEE Trans. Power Del.*, vol. 32, no. 2, pp. 1098–1105, Apr. 2017, doi: 10.1109/TPWRD.2016.2598572.

[27] X. Wang and E. Foo, ''Assessing industrial control system attack datasets for intrusion detection,'' in *Proc. 3rd Int. Conf. Secur. Smart Cities, Ind. Control Syst. Commun. (SSIC)*, Oct. 2018, pp. 1–8, doi: 10.1109/SSIC.2018.8556706.

[28] Y. Kwon, H. K. Kim, Y. H. Lim, and J. I. Lim, ''A behavior-based intrusion detection technique for smart grid infrastructure,'' in *Proc. IEEE Eindhoven PowerTech*, Jun. 2015, pp. 1–6, doi: 10.1109/PTC.2015.7232339.

[29] H. Yoo and T. Shon, ''Novel approach for detecting network anomalies for substation automation based on IEC 61850,'' *Multimedia Tools Appl.*, vol. 74, no. 1, pp. 303–318, 2015, doi: 10.1007/s11042-014-1870-0.

[30] *Communication Networks and Systems for Power Utility Automation—Specific Communication Service Mapping (SCSM)—Mappings to MMS*, document IEC 61850-8-1:2011, IEC, 2011.

[31] E. A. Leal Piedrahita, ''Hierarchical clustering for anomalous traffic conditions detection in power substations,'' *Ciencia Ingeniería Neogranadina*, vol. 30, no. 1, pp. 75–88, Nov. 2019, doi: 10.18359/rcin.4236.

[32] T. S. Ustun, S. M. S. Hussain, A. Ulutas, A. Onen, M. M. Roomi, and D. Mashima, ''Machine learning-based intrusion detection for achieving cybersecurity in smart grids using IEC 61850 GOOSE messages,'' *Symmetry*, vol. 13, no. 5, p. 826, May 2021, doi: 10.3390/sym13050826.

[33] X. Wang, C. Fidge, G. Nourbakhsh, E. Foo, Z. Jadidi, and C. Li, ''Feature selection for precise anomaly detection in substation automation systems,'' presented at the 13th IEEE PES Asia–Pacific Power Energy Eng. Conf., Kerala, India, 2021.

[34] T. Zoppi, A. Ceccarellli, and A. Bondavalli, ''An initial investigation on sliding windows for anomaly-based intrusion detection,'' in *Proc. IEEE World Congr. Services*, Milan, Italy, Mar. 2019, pp. 99–104, doi: 10.1109/SERVICES.2019.00031.

[35] P.-P. De Breuck, G. Hautier, and G.-M. Rignanese, ''Materials property prediction for limited datasets enabled by feature selection and joint learning with MODNet,'' *NPJ Comput. Mater.*, vol. 7, no. 1, p. 83, Jun. 2021, doi: 10.1038/s41524-021-00552-2.

[36] H. Qiu, Y. Tu, and Y. Zhang, ''Anomaly detection for power consumption patterns in electricity early warning system,'' in *Proc. 10th Int. Conf. Adv. Comput. Intell.*, Xiamen, China, 2018, pp. 867–873, doi: 10.1109/ICACI.2018.8377577.

[37] H. M. Gomes, J. Read, and A. Bifet, ''Machine learning for streaming data: State of the art, challenges, and opportunities,'' *ACM SIGKDD Explor. Newslett.*, vol. 21, no. 2, pp. 6–22, Feb. 2019, doi: 10.1145/3373464.3373470.

[38] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, ''Supervised machine learning: A review of classification techniques,'' in *Emerging Artificial Intelligence Applications in Computer Engineering*, vol. 160, I. Maglogiannis, K. Karpouzis, B. A. Wallace, and J. Soldatos, Eds. The Netherlands: IOS Press, 2007, pp. 3–24.

[39] C. Azad and V. K. Jha, ''Decision tree and genetic algorithm based intrusion detection system,'' in *Proc. 2nd Int. Conf. Microelectron., Comput. Commun. Syst.* Singapore: Springer, 2019, pp. 141–152.

[40] K. Chen, R. J. Mahfoud, Y. Sun, D. Nan, K. Wang, H. Haes Alhelou, and P. Siano, ''Defect texts mining of secondary device in smart substation with GloVe and attention-based bidirectional LSTM,'' *Energies*, vol. 13, no. 17, p. 4522, Sep. 2020, doi: 10.3390/en13174522.

[41] C. Yin, L. Xia, and J. Wang, ''Application of an improved data stream clustering algorithm in intrusion detection system,'' in *Advanced Multimedia and Ubiquitous Engineering, FutureTech*, J. J. Park, S.-C. Chen, K.-K. R. Choo, Eds. Singapore: Springer, 2017, pp. 626–632.

[42] H. Lahza, K. Radke, and E. Foo, ''Applying domain-specific knowledge to construct features for detecting distributed denial-of-service attacks on the GOOSE and MMS protocols,'' *Int. J. Crit. Infrastruct. Protection*, vol. 20, pp. 48–67, Mar. 2018, doi: 10.1016/j.ijcip.2017.12.002.

[43] I. A. Khan, A. Akber, and Y. Xu, ''Sliding window regression based short-term load forecasting of a multi-area power system,'' in *Proc. IEEE Can. Conf. Electr. Comput. Eng.*, Edmonton, AB, Canada, 2019, pp. 1–5, doi: 10.1109/CCECE.2019.8861915.

[44] A. Shamisa, B. Majidi, and J. C. Patra, ''Sliding-window-based real-time model order reduction for stability prediction in smart grid,'' *IEEE Trans. Power Syst.*, vol. 34, no. 1, pp. 326–337, Jan. 2019, doi: 10.1109/TPWRS.2018.2868850.

[45] D. Lu, X. Liao, F. Xu, and J. Bai, ''Anomaly detection method for substation equipment based on feature matching and multi-Semantic classification,'' in *Proc. 6th Asia Conf. Power Electr. Eng.*, Chongqing, China, 2021, pp. 109–113, doi: 10.1109/ACPEE51499.2021.9437096.

[46] X. An, F. Lin, and L. Yang, ''Node state monitoring scheme in fog radio access networks for intrusion detection,'' *IEEE Access*, vol. 7, pp. 21879–21888, 2019, doi: 10.1109/ACCESS.2019.2899017.

[47] W. Wu, Y. Huang, R. Kurachi, G. Zeng, G. Xie, and R. Li, ''Sliding window optimized information entropy analysis method for intrusion detection on in-vehicle networks,'' *IEEE Access*, vol. 6, pp. 45233–45245, 2018, doi: 10.1109/ACCESS.2018.2865169.

[48] G. Baldini, ''On the application of entropy measures with sliding window for intrusion detection in automotive in-vehicle networks,'' *Entropy*, vol. 22, no. 9, p. 1044, Sep. 2020, doi: 10.3390/e22091044.

[49] Z. Ding and M. Fei, ''An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window,'' *IFAC Proc.*, vol. 46, no. 20, pp. 12–17, 2013, doi: 10.3182/20130902-3-CN-3020.00044.

[50] M. Ahmed and A.-S.-K. Pathan, ''False data injection attack (FDIA): An overview and new metrics for fair evaluation of its countermeasure,'' *Complex Adapt. Syst. Model.*, vol. 8, no. 1, pp. 1–14, Dec. 2020, doi: 10.1186/s40294-020-00070-w.

**XUELEI WANG** received the M.Sc. degree in IT (security) from the Queensland University of Technology, in 2018, where he is currently pursuing the Ph.D. degree researching cyber-security issues in industrial control systems. His research interests include network and system security, malware analysis, network forensics, anomaly detection, and smart substation protection.

**COLIN FIDGE** is a Full Professor with the School of Computer Science, Queensland University of Technology, where he teaches programming fundamentals and research principles. His research interests include high-integrity software engineering, modelling and analysis of complex, computer-based systems, and business process modelling and analysis.

**GHAVAMEDDIN NOURBAKHSH** (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical engineering from California State University, Sacramento, USA, the M.Sc. degree in power system reliability from the University of Saskatchewan, Canada, and the Ph.D. degree from the Queensland University of Technology (QUT), Australia. He is currently an Academic Member with the School of Electrical Engineering and Robotics, QUT, and has actively been involved in research, teaching, and consulting to local utilities for more than 25 years. He has supervised multiple Ph.D. and master's students obtaining their research thesis degrees successfully, and has more than 1800 citations from his research publications based on Google Scholar. He has also been leading various research projects funded by local distribution and transmission utilities.

**ERNEST FOO** (Member, IEEE) is an Associate Professor with the School of Information and Communication Technology, Griffith University. He has published over 90 refereed papers, including 20 journal articles. His research can be broadly grouped into the field of applied cyber security. National critical infrastructure systems, such as electricity substations and water treatment plants are becoming more connected and are susceptible to cyber attacks. His research work looks for new ways to detect cyber attackers in critical infrastructure systems and prevent those attackers from being effective. His research interests include machine learning, data mining, cryptographic protocols, and network simulations.

**ZAHRA JADIDI** received the Ph.D. degree in network security from the School of Information and Communication Technology, Griffith University. She is a Research Fellow in cybersecurity at the Queensland University of Technology. Her research interest is the security of critical infrastructure, anomaly detection, and automation of security analysis.

**CALVIN LI** received the M.Sc. degree in internetworking from the University of Technology Sydney. He is a Principal Engineer with Instrumentation, Electrical and Control Systems, former Manager SCADA and Operations Technology Security at Jemena Ltd. He is a Charted Professional Engineer, register both nationally and in Queensland. His interest is to apply the research outcome into practical use within the critical infrastructure business to protect assets from cybersecurity risks.