international journal of critical infrastructure protection 49 (2025) 100762

Contents lists available at ScienceDirect

International Journal of Critical Infrastructure Protection

journal homepage: www.elsevier.com/locate/ijcip

STADe: An unsupervised time-windows method of detecting anomalies in oil and gas Industrial Cyber-Physical Systems (ICPS) networks

Abubakar Sadiq Mohammed *, Eirini Anthi, Omer Rana, Pete Burnap, Andrew Hood

School of Computer Science, Cardiff University, UK

ARTICLE INFO

Keywords:
Cybersecurity
Anomaly detection
Operational technology
Cyber-physical systems
Time windows

ABSTRACT

Critical infrastructure and Operational Technology (OT) are becoming more exposed to cyber attacks due to the integration of OT networks to enterprise networks especially in the case of Industrial Cyber-Physical Systems (ICPS). These technologies that are a huge part of our daily lives usually operate by having sensors and actuators constantly communicating through an industrial network. To secure these industrial networks from cyber attacks, researchers have utilised misuse detection and Anomaly Detection (AD) techniques to detect potential attacks. Misuse detection methods are unable to detect zero-day attacks while AD methods can, but with high false positive rates and high computational overheads. In this paper, we present STADe, a novel Sliding Time-window Anomaly Detection method that uses a sole feature of network packet inter-arrival times to detect anomalous network communications. This work aims to explore a mechanism for detecting breaks in periodicity to flag anomalies. The method was validated using data from a real oil and gas wellhead monitoring testbed containing field flooding, SYN flooding, and Man-in-the-Middle (MITM) attacks — which are attacks that are popularly used to target the availability and integrity of oil and gas critical infrastructure. The results from STADe proved to be effective in detecting these attacks with zero false positives and F1 scores of 0.97, 0.923, and 0.8 respectively. Further experiments carried out to compare STADe with other unsupervised machine learning algorithms – KNN, isolation forest, and Local Outlier Factor (LOF) – resulted in F1 scores of 0.55, 0.673, and 0.408 respectively. STADe outperformed them with an F1 score of 0.933 using the same dataset.

1. Introduction

In modern times, our daily lives have become increasingly dependent on Industrial Cyber-Physical Systems (ICPS). Whether it is monitoring and controlling an efficient railway network, or controlling hazardous hydrocarbons being transported within oil and gas pipelines, ICPS are managing to keep our environments safe and predictable. However, innovation through digitisation and automation has exposed these systems to cyber threats [1]. This threat from cyber-attacks has attracted increased interest from researchers in academia and industry to determine the best ways to protect our critical infrastructure.

ICPS are typically composed of three control components (i) Programmable Logic Controllers (PLC), (ii) Supervisory Control and Data Acquisition (SCADA), and (iii) Distributed Control Systems (DCS). However, one dominant component throughout all industrial control systems is the communication network which is responsible for connecting all equipment and devices by electrical interfaces and communication protocols to ensure all systems communicate efficiently [2]. It is gradually becoming evident that securing industrial network communications from cyber-attacks should be one of the key steps in ensuring that critical infrastructure is protected.

Network traffic from industrial networks exhibits strong periodic patterns [3]. This work is motivated by the observation that industrial control systems and PLC-based systems have a high degree of periodicity in behaviour when used in a real-world context compared to other Information Technology (IT)-based traffic. This is because, rather than traffic being generated mainly from random user-generated workflows – as in the case of enterprise/IT networks – industrial network traffic is primarily generated from the consistent polling of data between systems with the aim of monitoring and controlling the process. This gives it a high repeatability resulting in a consistent pattern. This could be likened to harmonious music, where each industrial network has its own tune represented as a pattern. This pattern is a basic representation of the behaviour of the industrial network under normal operations.

Having such high periodicity has its advantages. One such advantage is that, if properly represented and modelled, any slight deviation

* Corresponding author.
E-mail addresses: mohammedas@cardiff.ac.uk (A.S. Mohammed), anthies@cardiff.ac.uk (E. Anthi), ranaof@cardiff.ac.uk (O. Rana), burnapp@cardiff.ac.uk (P. Burnap), hooda3@cardiff.ac.uk (A. Hood).

https://doi.org/10.1016/j.ijcip.2025.100762
Received 6 February 2023; Received in revised form 7 January 2025; Accepted 8 April 2025
Available online 23 April 2025

from the established basic network pattern could be easily identified and flagged as anomalous behaviour. This regularity of patterns present in industrial control network traffic makes anomaly detection very promising [4].

Intrusion detection Systems (IDS) can be classified broadly into misuse detection (signature/rule-based Intrusion Detection Systems) and anomaly detection [5]. Historically, misuse IDS have proven to be effective in identifying traditional (known) cyber attacks that indicate discriminate patterns [6–9], but in spite of this, these IDS are less effective at detecting zero-day attacks that utilise novel methods of exploiting vulnerabilities with persistence. This is an advantage that anomaly detection methods have because they are more capable of detecting novel anomalous scenarios. However, despite this advantage, anomaly detection solutions are not commonly applied in practice because of high computational overheads and high false-positive rates [10] - often leading to a high number of false alarms overwhelming security experts with alerts [11]. This is relevant, since the usefulness of intrusion detection systems is greatly influenced by the false-positive rate [12].

With the increasing frequency of zero-day attacks being carried out on critical infrastructure [13], it has become evident that to improve widespread adoption, anomaly detection methods need to be improved upon to reduce the high false-positive rates and computational complexities. This would potentially increase the protection levels of critical infrastructure against cyber threats.

In this study, we present STADe - a Sliding Time-window Anomaly Detection method that uses a sliding window to characterise the periodicity of any given industrial network using the packet timings to create a mini-model of the system which represents the normal operation pattern. This periodic pattern allows the use of anomaly detection to determine where the periodicity is broken (e.g. injection of cyber-attacks). This work is focused on a computationally efficient mechanism to identify this break in periodicity, which flags anomalies and detects cyber-attacks. Previous studies on industrial network anomaly detection have mostly focused on inspecting individual packets through deep packet inspection (DPI) or other custom modules which can be error-prone – as evidenced by the high false positive rates experienced by these methods. The advantage of using a time window, comprising multiple packets, is that it can adequately capture network behaviour over a specified period of time, and thus, is more effective at labelling a particular series of packets as either normal or anomalous with a much lower false positive rate when compared with trying to label a specific individual packet.

The main contributions of this paper are:

- A novel method of defining the periodicity of any given industrial network by using the packet timings to create a mini-model of the system representing the normal operation pattern.
- A mechanism to detect a break in periodicity, which is used to flag anomalies by use of a sliding time window.
- A method to visualise the periodicity of an industrial network using only packet inter-arrival times represented as points in 3-dimensional space.
- Four labelled datasets collected from an oil and gas industrial testbed containing field flooding attacks, SYN flooding attacks, and Man-in-the-Middle (MITM) attacks.

The remainder of this paper is structured as follows: Section 2 discusses the related work in this research area. Section 3 describes the approach and STADe methodology. In Section 4 description of the experiments carried out is given, while Section 5 presents the results and discusses them in more detail. Section 6 compares STADe with other ML methods and the conclusion is in Section 7.

2. Related work

Despite the importance of critical infrastructure, and the increasing threats to it from cyber attacks, only a few studies have investigated anomaly detection methods specifically for industrial networks. Many current network anomaly detection systems are based on supervised machine learning methods which are often expensive and difficult to obtain training data [5,14], while unsupervised machine learning anomaly detection methods are not widely used in practice because of high false positive rates that tend to overwhelm security analysts with false alerts. To combat these problems, researchers have recently turned to deep learning techniques, which have also increased computational overheads. Industrial control networks operate in resource-constrained environments and require lightweight solutions. For these reasons, we have not considered machine learning and deep learning-based anomaly detection methods.

In other recent anomaly detection approaches for industrial networks, such as [15], the authors used a timing-based anomaly detection system that uses the statistical attributes of the communication patterns. Their proposed intrusion detection system identifies unique sets of request–response events from request types and requested addresses. Jiang et al. [16] also proposed a method to detect network traffic anomalies by using a sliding window that uses Decomposable Principal Component Analysis (DPCA) to handle network traffic signals. The DCPA handles the traffic of all original destination flows in a network while an adaptive clique division enables it to dispose of the dynamic network traffic. The method comprises utilising compressed features of the network traffic including byte size to classify anomalous scenarios. Their work was evaluated using traffic data from the Abilene network as ground traffic. By incorporating addresses into their learning modules, the models in [15,16] would struggle to detect stealthier attacks like MITM which spoofs legitimate IP addresses morphed into the data stream.

In [17], Tekeoglu et al. used network traffic features to capture the system-specific anomalies. They did this by extracting a number of packet-based features from pcap files using 3 s time windows which included average bytes in the window, average seconds between each consecutive packet in a window, and unique destination IP addresses in the window amongst others. This method required constant iterations between network traffic features to determine the most appropriate metric which increases its computational overhead.

The authors in [4,18] also tried to utilise the traffic periodicity in industrial networks by using message repetition and timing information to automatically learn traffic models that capture periodic patterns. The authors in [4] proposed a period analyser composed of three modules: (i) Multiplexer, (ii) Tokeniser, and (iii) Learner which worked in a sequence comprising preprocessing the network traffic and separating it into different flows, transforming each packet into a protocol-independent format called a token, and finally processing each token to identify and characterise periodic activities called cycles. Their method involves filtering and grouping packets based on server address, IP protocol, server port, and client address. One key limitation of their study is that the tokeniser and multiplexer modules need to be adapted in order to accommodate new industrial protocols as it was designed for Modbus and MMS protocols. The authors in [18] also relied on the stable and persistent control flow communications in industrial networks to develop a fingerprinting methodology to capture normal behaviour characteristics. They extracted multiple features such as packet arrival order, packet size, direction, and inter-arrival time to classify network behaviour. This requirement of using multiple features from network traffic increases the computational head on the system.

In general, industrial network anomaly detection requires a lightweight solution because it operates in a resource-constrained environment. The more features the IDS model utilises in its detection engine, the higher the computational overhead on the system. All of the approaches mentioned here utilise multiple features from network traffic

Table 1
Summary of related work.

Reference	Timing-based	Window-based	Single feature-based	Zero false positives
[15]	●			
[16]	●	●		
[17]		●		
[4]	●			
[18]	●			
STADe	●	●	●	●

in order to train a learner module that would subsequently classify the packet as normal or otherwise. By using packet inter-arrival times as the only feature, STADe offers a method that has less computational head that can operate in a resource-constrained environment. The timing feature also remains unchanged even after encryption. This gives STADe an advantage as it can be deployed alongside other encryption-based security solutions to improve protection. This is usually a pitfall for most anomaly detection engines as they require visibility into the network data (e.g. source/destination addresses, source/destination ports, and data payloads). This is all summarised in Table 1.

In the next section, we describe the STADe approach and methodology in detail, showing how it could be used to characterise and subsequently detect industrial network traffic anomalies.

3. STADe: Approach and methodology

3.1. Measuring periodicity

To identify breaks in periodicity in a given industrial network, we first have to be able to measure network traffic periodicity. This involves the use of data in the form of ordered observations with respect to time. To increase ease of computation, some features of the data may be neglected, and thus only analysing the time between events [19]. If only the time of the event (i.e. data/packet transmission) and no further details are stored, the event sequence is called a point sequence [19]. Several studies, such as [20], have done this by capturing number of packets per second as a feature, while others (e.g. [21]) used arrival time of packets as its core feature. This not only underscores the importance that packet timing, observed as point sequences, has in defining the periodic nature of industrial network traffic but also, shows its usefulness in getting information from network traffic [22].

Hubballi et al. [23] posited that periodic communications exhibit very low variance and standard deviation considering their inter-time differences and determined this low variance by taking the standard deviation of packet inter-arrival times between network packets. We adapted this approach in our study by calculating the standard deviation of packet inter-arrival times within a time window of packets.

Thus, the standard deviation of a window (SD_w) gives the variance or level of dispersion within that distribution. This is for any given window with each element in the distribution (x_i), sample mean ($\bar{x}$), and window size (n).

$$SD_w = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}} \tag{1}$$

3.2. Detecting deviation from periodicity

The advantage of being able to easily measure the periodicity of an industrial network is to gain the ability to detect a break or deviation from the periodic pattern. To do this, the data feature collected (i.e. packet inter-arrival times) can be segmented into time windows of the same size. These time windows can then be compared to one another to determine their similarity (normal traffic) or dissimilarity (anomaly). We later describe a metric – the diff score – which will compute if the data is anomalous or not. One way to do this is to select a representative time window (baseline/sliding time window), which can then be compared to all other windows using some distance function. The authors in [24] did this by computing the average trend in a segment that minimises the sum of distances to the other segments — in other words, computing the euclidean distances. In our study, we have adopted this approach to compute the euclidean distance between the average trend of a time window distribution and that of the sliding window under consideration.

This is achieved by computing the L2-norm as it is the estimate of location that minimises the euclidean distance between two time windows. This is represented as the diff centre (DC_w) for a given window (i), window size (n), and sliding window (j).

$$DC_w(i,j) = \sqrt{\Sigma_{i=1}^{n}(i-j)^2} \tag{2}$$

3.3. Visualising periodic behaviour

Recording the datastream from an industrial network as point events allows us to represent the packet inter-arrival times visually as points in 3-dimensional space, similar to the approaches in [22,25] where the authors try to classify network traffic visually. In [22], bigrams of packets are visualised where the coordinates are defined by the first packet's size (X), the inter-arrival time between the two packets (Y), and the second packet's size (Z), while the coordinates in [25] represent sequence number, frame length, and packet number to classify telecommunications traffic.

The STADe visualisation approach differs from previous studies because each coordinate represents information from only the packet inter-arrival times between three consecutive packets. The advantage of using 3-dimensional spaces over 2D is that just one coordinate can be used to display information of more packets (i.e. 3 packets) and would require less space to represent a network's basic pattern than a 2-dimensional space would.

The packet flow regularity in industrial networks is the characteristic that enables a sliding window with a single feature of packet timings to represent the basic normal pattern of operations in a given industrial system. This allows us to compute any significant deviations from the basic pattern represented in the sliding window to detect anomalous scenarios within the network.

3.4. Methodology

Based on the described approach, the framework of the STADe methodology (shown in Fig. 1) is described as follows:

1. Extract packet inter-arrival times δ from a Cyber-Physical System data stream as a vector.
2. Divide the extracted δ into windows (W) of same segment size, n, such that $\delta = (W_1, W_2, \ldots, W_m)$ with elements $t_i \in W$. A fixed segmentation $\delta(n)$ of size n is a division of δ into m windows where each of the windows consists of n consecutive elements from δ.
3. Select baseline/sliding window containing normal traffic.
4. Calculate the standard deviation and diff centres for each window.
5. Compare the sliding window with other windows by computing the diff score S. The diff score is a combination of the standard deviation and diff centres using a weight x ranging from 0–1. This essentially allows the flexibility of assigning more weight

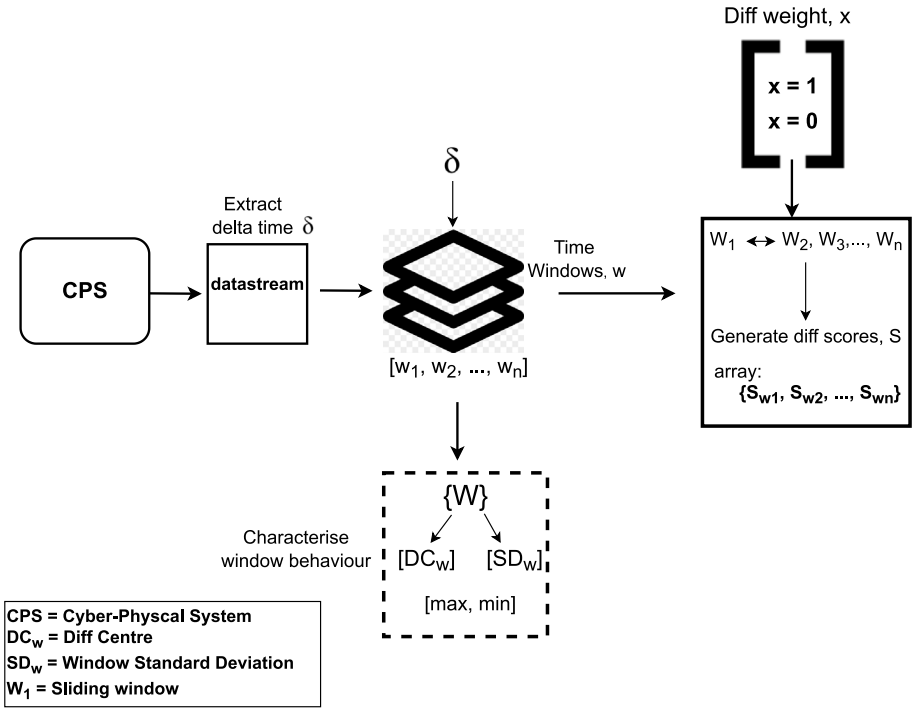


Fig. 1. STADe methodology.

(importance) to either the standard deviation or diff centre (i.e. weight tends towards 0 or 1 respectively) or assigning equal importance to both (i.e. weight = 0.5).

6. The diff scores of all window segment comparisons $S_\delta = (S_{w_1}, S_{w_2}, \ldots, S_{w_n})$ is generated and stored as an array.

In the next section, a detailed description of the experiments carried out to validate the STADe methodology is given with emphasis on the hyper-parameters, dataset collection, and visualisation methods.

4. Experiments

4.1. Dataset collection

For these experiments, a total of four (4) datasets were collected from the wellhead monitoring testbed described in [26] and labelled to enable proper evaluation of the performance of STADe on detection of each attack. All datasets were designed to have a similar attack time pattern where the first 80% represents normal traffic, an attack is executed in the next 10%, and the final 10% of the network traffic is normal as illustrated in Fig. 2. This helps us to have a general idea of where our model should be detecting attacks and helps with evaluating its performance.

The attacks considered for these experiments were derived from the most frequent attacks on oil and gas critical infrastructure targeting "Availability" and "Integrity" [27]. For attacks targeting availability, field flooding attacks and SYN flooding attacks were carried out while for attacks targeting integrity, a Man-in-the-Middle (MITM) attack was executed. It is important to note that a successful MITM attack could serve as an initial phase which makes a number of further attacks possible in a second phase. A list of these possible second phase attacks are highlighted in Table 2. The summary of all the datasets is also shown in Table 3

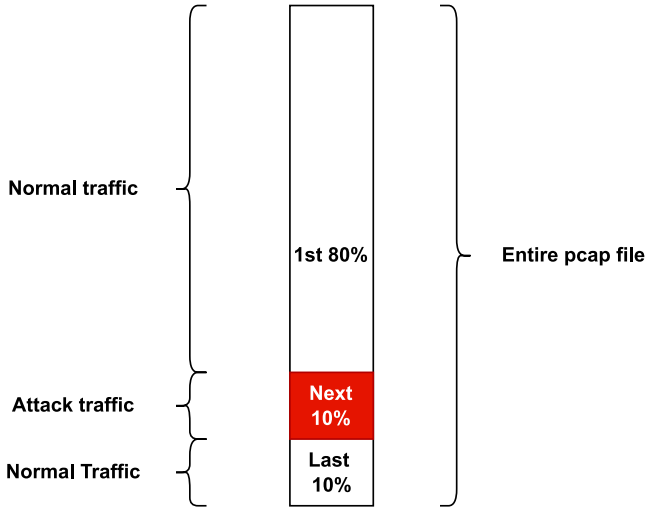


Fig. 2. Dataset creation.

4.2. Visualising entire datasets

The initial phase of the experiments entailed making 3D plots of the network packets under normal operations to visually investigate if there is a set pattern that could represent network behaviour. To do this, normal traffic data was collected from the testbed over a 22 h period. The resulting dataset contained 3,409,005 packets. A second dataset was collected from the testbed comprising just 3.8 h of normal traffic + field flooding attack [26]. This second dataset contained 472,887 packets. Both plots are shown in Fig. 3.

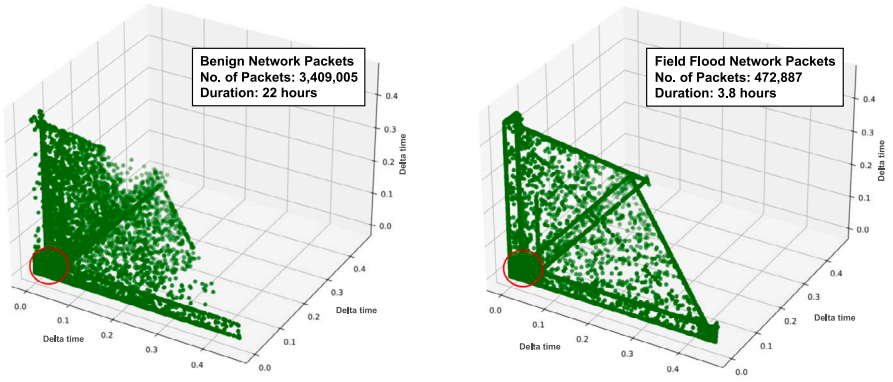
From Fig. 3, a pattern is observable, however, it seems distorted with a lot of noise. This is the primary limitation of plotting an entire dataset on a single plot because over 90% of the data points lie within

Table 2
Further potential attacks after successful MITM attack.

1st phase Attack	2nd Phase Attack	Impact
MITM	Data interception	Attacker can eavesdrop on communication between HMI and PLC, capturing sensitive commands and process state
	Data tampering	Attacker can modify data transmitted to PLC, altering commands
	Session hijacking	Attacker can hijack an established session between two parties and taking control by impersonating one of the legitimate parties
	Credential theft	Admin login credentials could be intercepted, enabling lateral movement within a network
	Replay attacks	Attacker can capture data packets and replay them later, potentially sending same commands but in the wrong context
	Malware delivery	Attacker can utilise MITM position to deliver malicious software into victim's network or devices
	DNS spoofing	Attacker can manipulate DNS responses, or intercept DNS queries within the network,potentially redirecting requests to malicious end points
	Phishing	After intercepting communication, the attacker can launch further phishing campaigns with targeted information. An example could be to target remote engineers accessing process systems

Table 3
Summary of datasets collected from testbed.

Dataset	Attack	Attack type	Target	Attack duration	Dataset duration	Total No. Pkts
Dataset 1 (normal traffic)	N/A	N/A	N/A	N/A	22.4 h	3,409,005
Dataset 2	Field flooding	DoS	Availability	1 h	3.8 h	472,887
Dataset 3	SYN flooding	DDoS	Availability	13 s	1.4 h	230,409
Dataset 4	MITM	See Table 2	Integrity	5.5 min	2.1 h	319,906



(a) 24-hour normal operations Network traffic - Packet inter-arrival times

(b) 3.8-hour normal+field flooding attack Network traffic - Packet inter-arrival times

Fig. 3. 3D plots for Network traffic packet inter-arrival times.

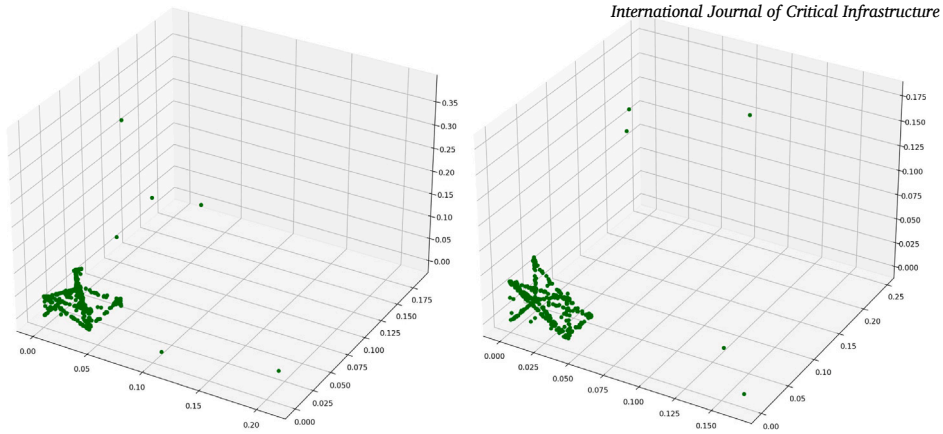
the dense circled area. As a result, the scale of the plot is larger than it needs to be to accommodate anomalous coordinates. It becomes evident that, to see deeper underlying patterns that could potentially lie within the dense circled area, a smaller-scaled plot representing a window (or subset) of network traffic would be more beneficial. This confirms the advantage of using the STADe methodology described in Section 3.4 to create a smaller sliding window, compare it with other windows and generate the diff scores between them.

Despite this limitation, the plot is still useful because when visually inspecting Figs. 3(a) and 3(b), there are some obvious distortions in the network pattern seen in 3(b) that were not evident in 3(a). These differences could be as a result of the field flooding attack but cannot be confirmed visually. In this case, having a numerical score as a basis for comparison and evaluation would be more beneficial. This numerical score is represented by the diff score described in the STADe methodology.

4.3. Hyper-parameters:

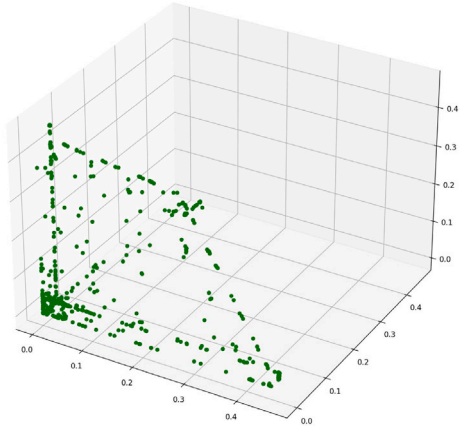
The next phase of the experiments to be carried out is to generate diff scores for all 4 datasets with the following hyper-parameters:

- Window size, n: Industrial networks have communication cycles (i.e. query-response-acknowledgement cycle). Each device is queried sequentially at least once until all devices have responded. Then the communication would loop and start all over.



(a) Dataset 2 baseline window - Packet inter-arrival times

(b) Dataset 2 random window - Packet inter-arrival times



(c) Field Flooding attack anomalous window 163 - Packet inter-arrival times

Fig. 4. 3D plots for Dataset 2 windows showing (a) Baseline normal traffic, (b) normal traffic, (c) Field flooding attack traffic.

To determine an adequate window size, a significant amount of loops (repetitions) containing all normal operating conditions should be captured within it. In this study, we investigated window sizes containing 30, 60, 90, and 120 s of network communication. An optimal window size of 60 s (1 min of network communication) was determined empirically. This contained approximately 2500 packets (100 repetitions) and more significantly, captured all communicating devices and their respective commands. While determining the window size, it was observed that below 60 s, the window did not capture enough information to characterise network behaviour, hence this resulted in some normal operations being flagged as anomalous (i.e. more false positives). Similarly, above 60 s, the window size contained more information than was necessary, which resulted in some anomalous behaviours being classified as normal (i.e. more false negatives).

It is important to note that this approach may vary in larger and more complex industrial environments.

- Diff Score weight: The diff score weight, ranging from 0–1, is the importance given to measurements of diff centres and areas of coverage. A diff weight of 0.5 was chosen to give equal importance to either parameter.
- Threshold: The next important parameter is the definition of a threshold. To define a suitable threshold, diff scores need to be generated using normal traffic data. The aim is to capture all possible usual network scenarios which include random packet retransmissions, legitimate connection resets, etc. The maximum diff score generated from normal traffic can be used as a valid threshold. The amount of normal traffic required would vary depending on the size of the industrial network. In our case, we monitored normal network traffic (with no attacks) for 22 h and collected the data. Diff scores were generated for the whole dataset containing normal traffic and the maximum diff score was selected as follows:

$$maxdiffscore_{normaloperations} = 0.00216024 = threshold$$

In this study, a static threshold is employed to ensure simplicity and focus on validating the proof-of-concept. This allows for clear interpretation of results and reproducibility, which are critical for establishing the feasibility of the proposed approach.

After defining the hyper-parameters, the first window (i.e. first 2500 packets), which had already been pre-determined to be a normal traffic window, was selected as the baseline window (i.e. sliding window). Recall that all subsequent windows will be compared with the baseline window, and a diff score generated to measure the differences. Any diff score (difference) above our determined threshold would be flagged as an anomaly. In our case, any window within the first 80% (or last 10%) of the dataset could have been chosen as our baseline window, however, we decided to use the first. This decision does not affect the results so the user can be flexible in their choice of a baseline/sliding window as long as it is a normal traffic window.

4.4. Evaluation metrics

To evaluate the performance of STADe on our labelled datasets, the following metrics were chosen:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{5}$$

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

$$FDR = \frac{FP}{FP + TP} \tag{7}$$

Where:
TP = True Positive
FP = False Positive
TN = True Negative
FN = False Negative
FPR = False Positive Rate
FDR = False Discovery Rate

The false positive rate is the more commonly used metric in the literature. However, in practice, the false discovery rate is what operators are more concerned with because it is more easily computed in operational environments. The FPR can be misleading when the proportion of malicious instances is extremely low, as is the case in industrial network anomaly detection. For these reasons, in addition to the FPR, we would also compute the false discovery rate.

The results of the generated diff scores for our attack datasets will be discussed in the next section.

5. Results

Diff scores were generated for all three (3) attack datasets using the hyper-parameters determined in Section 4. We discuss each attack separately in the following subsections.

5.1. Selection and visualisation of baseline window

The first step was to visualise our baseline window. This becomes our sliding window and a sort of label for the dataset that would be compared with every other window by generating a diff score. It can also serve as a basis for a visual comparison with any other window identified as having a diff score higher than the set threshold. As an example, we compared this baseline window to another normal traffic window and generated a diff score between them to establish a correlation. Our baseline window indexes were 0–2499 (i.e. first 2500 packets) while the random window comparison indexes were x - (x + 2499), where x = 6000 (i.e. the 6000th packet). This procedure was repeated for all three datasets containing attacks and the results obtained are discussed in the next subsections.

5.2. Diff score generation for Dataset 2 (Field flooding attack)

The baseline window and random window plots are shown in Figs. 4(a) and 4(b) respectively while a diff score of **0.000698620** was generated from their comparison. Visually inspecting both plots (i.e. Figs. 4(a) and 4(b)), a similar pattern can be observed, although, it is not conclusive. However, their similarity, when evaluated mathematically using the diff score, was confirmed as the result was significantly below the threshold of **0.00216024**. This method confirms that the random window (i.e. packet index 6000–7499), when compared with the baseline window contains normal traffic.

Next, the diff scores were generated for the entire dataset, culminating in a total of 185 windows (i.e. 185 diff scores). Using our pre-determined threshold, 16 windows with diff scores higher than the threshold were identified as shown in Fig. 5. A summary of the anomalous windows and their diff scores is represented in Table 4a.

To confirm our findings visually, a plot of any of the anomalous windows (e.g. window 163) was created (see Fig. 4(c)) and it showed an obvious deviation from the baseline window pattern seen in Figs. 4(a) and 4(b). Finally, to evaluate the detection of the field flooding attack using the diff score methodology, an F1 score of 0.97 was obtained.

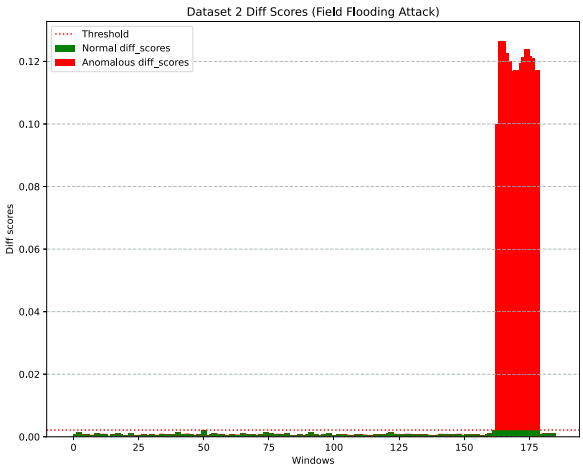


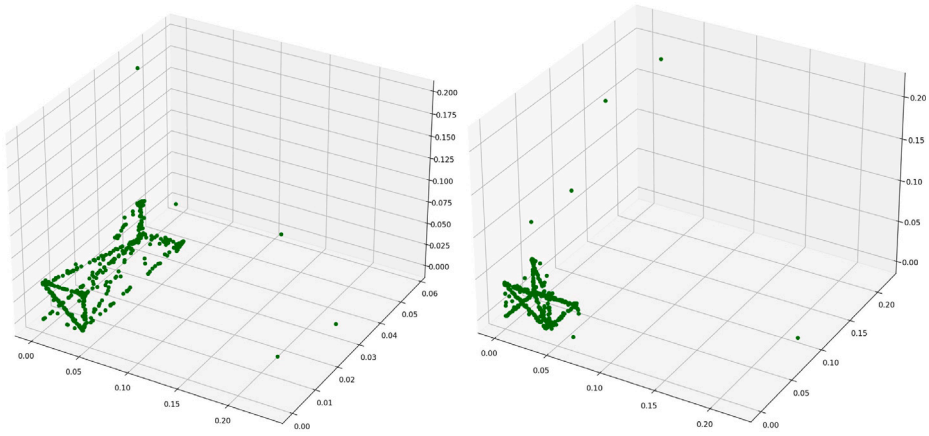
Fig. 5. Diff Scores for Dataset 2 - Field flooding attack.

Table 4
Anomalous windows with diff scores higher than set threshold (a) Dataset 2, (b) Dataset 3, (c) Dataset 4.

(a) Field flooding attack	
Threshold = 0.00216024	
Anomalous windows	Diff Score
Window 163	0.099775081
Window 164	0.12644391
Window 165	0.126265512
Window 166	0.12266697
Window 167	0.119892574
Window 168	0.116689842
Window 169	0.113587351
Window 170	0.116997225
Window 171	0.114008493
Window 172	0.119197875
Window 173	0.121269935
Window 174	0.123674147
Window 175	0.121695499
Window 176	0.120863596
Window 177	0.112604565
Window 178	0.117216779
(b) SYN flooding attack	
Threshold = 0.00216024	
Anomalous windows	Diff Score
Window 79	0.029395
Window 80	0.029166
Window 81	0.029152
Window 82	0.029189
Window 83	0.029192
Window 84	0.02324
(b) MITM attack	
Threshold = 0.00216024	
Anomalous windows	Diff Score
Window 119	0.002651
Window 122	0.012518

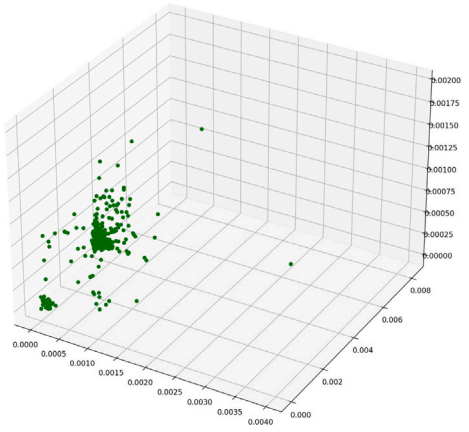
5.3. Diff score generation for Dataset 3 (SYN flooding attack):

The same methodology was applied to Dataset 3 with the SYN flooding attack. The baseline window and random window plots are shown in Figs. 6(a) and 6(b) respectively while a diff score of **0.00069579** was generated from their comparison. Again, with a visual inspection, a similarity of patterns is observable but can only be confirmed using the diff score. The generated diff score was also below the threshold of **0.00216024** which confirms that the random window (i.e. packet



(a) Dataset 3 baseline window - Packet inter-arrival times

(b) Dataset 3 random window - Packet inter-arrival times



(c) SYN Flooding attack anomalous window 79 - Packet inter-arrival times

Fig. 6. 3D plots for Dataset 3 windows showing (a) Baseline normal traffic, (b) normal traffic, (c) SYN flooding attack traffic.

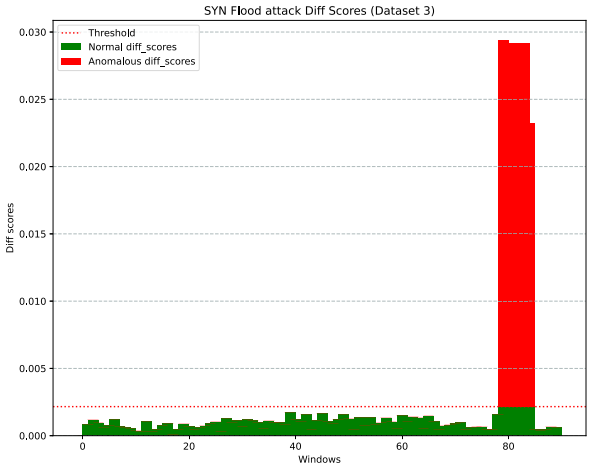


Fig. 7. Diff Scores for Dataset 3 - SYN flooding attack.

index 6000–7499), when compared with the baseline window is normal traffic.

Diff scores were generated for the entire dataset, which resulted in 90 windows/diff scores. Six windows were identified to have diff scores higher than the set threshold and are summarised in Fig. 7 and Table 4b.

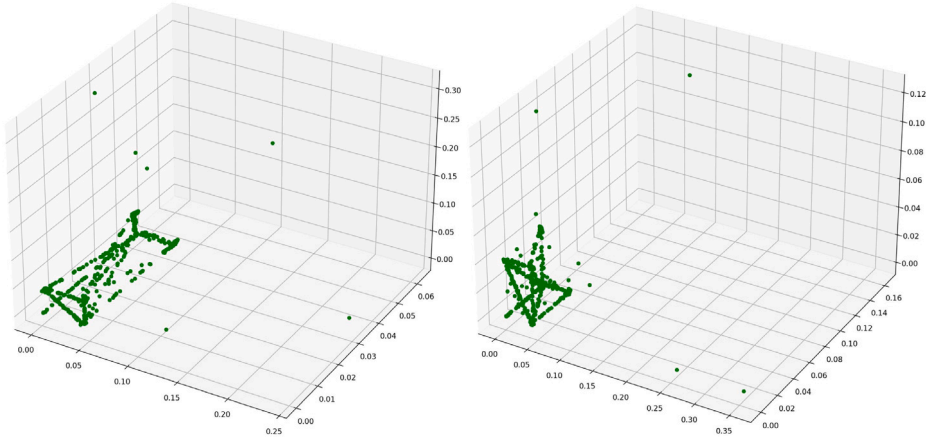
Again, to confirm our findings visually, a plot of any of the anomalous windows (e.g. window 79) was created (see Fig. 6(c)) and it showed a clear lack of similarity from both Figs. 6(a) and 6(b) which represent normal traffic patterns. Finally, to evaluate the detection of the syn flooding attack using the diff score methodology, an F1 score of 0.923 was obtained.

5.4. Diff score generation for dataset 4 (MITM attack):

For the final dataset containing MITM attacks, the same methodology was applied to select the baseline window and random window plots. A diff score of **0.00070328** was generated from their comparison. The window selected was confirmed to be normal traffic as the diff score fell below the set threshold.

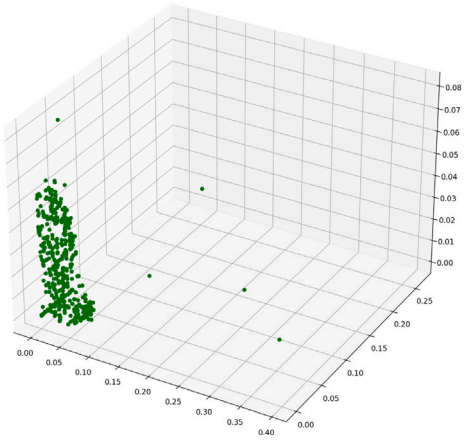
In the same manner, diff scores were generated for the entire dataset, which resulted in 125 windows/diff scores. For the MITM dataset, because it is a much stealthier attack, only two windows were identified to have diff scores higher than the set threshold and are summarised in Fig. 9 and Table 4c.

Finally, to confirm our findings visually, a plot of any of the anomalous windows (e.g. window 119) Fig. 8(c) was observed to be distinctively different in the pattern when compared both Figs. 8(a) and 8(b) which represent normal traffic patterns. However, when evaluating the detection of the MITM attack using the diff score methodology, an F1 score of 0.8 was obtained. The reason for the lower F1 score when compared with the previous field flooding and SYN flooding attacks is that the MITM is a stealthier attack that is mostly detected at two points: (a) when ARP poisoning begins, and (b) when the ARP table is



(a) Dataset 4 baseline window - Packet inter-arrival times

(b) Dataset 4 random window - Packet inter-arrival times



(c) MITM attack anomalous window 119 - Packet inter-arrival times

Fig. 8. 3D plots for Dataset 4 windows showing (a) Baseline normal traffic, (b) normal traffic, (c) MITM attack traffic.

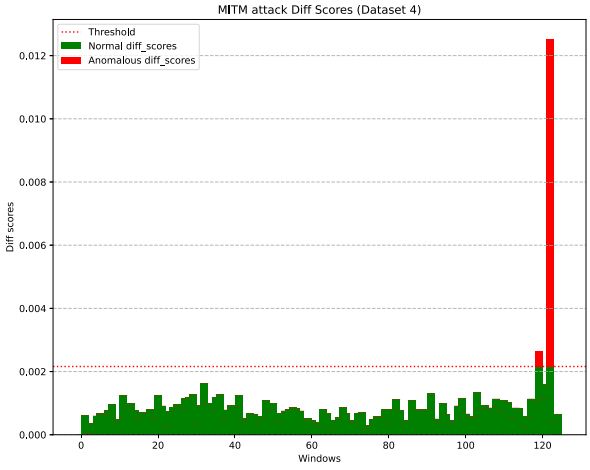


Fig. 9. Diff Scores for Dataset 4 - MITM Attack.

reverted to its original state. An interesting observation is that the ARP poisoning (the start of the MITM attack) began in window 118, but the diff score of that window is below the threshold. However, the diff score of the following window 119 was above the threshold. This may be due to the fact that the ARP poisoning packets occurred towards the tail end of window 118. However, it is still very interesting that the STADe methodology was able to observe a change in network traffic pattern in the next window 119. It did correctly flag window 122 as anomalous, which is when the ARP table was reverted to its original state, signifying the end of the MITM attack.

5.5. Summary of results

In summary, STADe was able to detect all the attacks by generating diff scores and having the right threshold setup for effective detection. The method proved to be effective in detection with no false positives recorded for any of the attacks. The field flooding attack had the highest F1 score of 0.97 and also had the highest number of anomalous windows (16). This was because the impact of the field flooding attack on the system lasted the longest. For the SYN flooding attack, an F1 score of 0.923 was achieved over 6 anomalous windows. Finally, for the MITM attack, the lowest F1 score of 0.8 was obtained. As explained earlier, this could be attributed to the fact that because the ARP poisoning occurred at the end of the window, the distortion was not significant enough for it to be detected as an anomaly, however, the detection occurred in the next window. One of the most important metrics regarding anomaly detection is the False Discovery Rate (FDR), which was zero for all attacks evaluated. This means that the STADe methodology was able to effectively measure the periodicity of industrial network traffic and also, segment the traffic into equally sized windows which were further compared with each other to detect deviations from normal patterns — in essence, anomalies. The summary of the results is highlighted in Table 5.

Table 5
Summary of STADe performance on the datasets.

Attack	Total windows	Anomalous windows	Precision	Recall	F1 score	FDR	FPR
Field flooding	185	16	1.0	0.94	0.97	0.0	0.0
SYN flooding	90	6	1.0	0.86	0.923	0.0	0.0
MITM	125	2	1.0	0.67	0.8	0.0	0.0

Table 6
Summary of combined dataset.

Attack	Attack type	Attack duration	Total capture duration	No. of Pkts
Field flooding	DoS	1 h		
SYN flooding	DDoS	13 s	7.3 h	1,023,202
MitM	Spoofing	5.5 min		

The selection of a window size of approximately 2500 packets (representing about one minute of network traffic) means that this tool can potentially detect anomalies within a minute of their occurrence.

With the results of the experiments showing the STADe methodology achieving zero false positives, it would be useful to analyse its effectiveness compared to current, mostly ML-based techniques. Within this group of anomaly detection methods (i.e. ML-based), unsupervised ML anomaly detection algorithms are most closely related to our chosen scenario. This will be explored in the next section.

6. Comparing STADe performance with unsupervised ML methods

Although in Section 2 the limitations of unsupervised ML methods in anomaly detection were articulated, it would still be useful to compare their performances and analyse STADe with the state of the art as most anomaly detection solutions employ ML methods. Amongst these, unsupervised ML anomaly detection algorithms are most closely related and could be applicable to industrial and operational scenarios (i.e. unlabelled data, resource-constrained environment). To achieve this, the same datasets would be utilised to keep the experiments as similar as reasonably practicable.

A number of unsupervised ML algorithms have been proposed in the literature for anomaly detection such as the graphical method, statistic method, distance-based method, density-based method, and model-based method [28]. Of these, the most frequently used are the distance-based method (e.g. K-Nearest Neighbour, KNN), density-based method (e.g. Local Outlier Factor, LOF), and the model-based method (e.g. isolation forest). This is mainly because of their ability to detect global and local (deeper lying) outliers especially when mapping high-dimensional data onto a low-dimensional subspace (as in the case of KNN and LOF) and also explicitly isolating anomalies rather than profiling normal instances (as in the case of isolation forest). As a result, these models perform well with high dimensional data with a low memory requirement. For these reasons, KNN, LOF, and isolation forest algorithms will be utilised in the experiments.

6.1. Experiments

Dataset Collection: The same datasets used in the experiments in Section 4 were also used in these experiments. The difference, however, was the elimination of Dataset 1 which was only useful for the STADe methodology to determine a suitable threshold. Also, datasets 2, 3, and 4 (containing field flooding, SYN flooding, and MITM attacks) were combined sequentially into a single dataset. This was done to enable a more concise analysis of results considering that multiple algorithms are being investigated. The combined dataset is summarised in Table 6. For effective comparison, the STADe methodology was also used to detect anomalies in this combined dataset.

Data Pre-Processing: All data pre-processing and feature selection methods used in this study were similar to those employed in [26] because the dataset was generated from the same testbed. This resulted in a dataset with 24 features. One notable difference in the data pre-processing approach adopted in this study is that there is no requirement for a train/test split. This is because anomaly detection works on the assumption that anomalous events are very rare, which in turn, produces highly imbalanced training datasets. As a result, the goal is to learn a valid model of the majority of data points (normal data) [29] which helps it detect deviations from the norm.

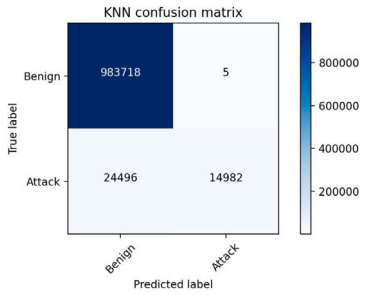
Furthermore, dimensionality reduction was applied to the dataset for the KNN and LOC experiments. KNN and LOC perform optimally when high-dimensional data is reduced and projected onto a lower-dimensional space. Therefore, Principal Component Analysis (PCA) was applied to reduce the features to a 2-dimensional array. This helps reduce the computational complexity required for detection. PCA is the most common dimensionality reduction technique [30]. By identifying directions of the highest variance from higher-dimensional data and projecting them onto a lower-dimensional subspace, when used with an ML model, it is able to reduce the number of parameters fed into the model without sacrificing important details in the data [31]. Default hyperparameters were retained in most cases except for the following empirically determined optimal choices:

- **KNN:** All default hyperparameter values
- **Isolation Forest:** n_estimators = 50, contamination = 0.048
- **LOF:** contamination: 0.048

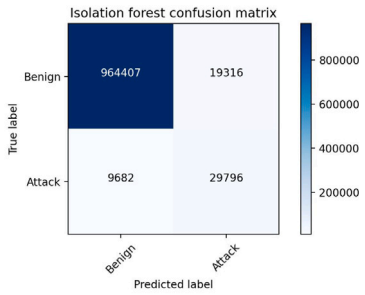
6.2. Results

The results of the experiments showed that the isolation forest algorithm had an F1 score of 0.673, with KNN and LOF having F1 scores of 0.55 and 0.455 respectively (Table 7). When compared on the same dataset, STADe recorded a higher F1 score of 0.933. Also, observing the FPR scores, all 3 ML algorithms recorded seemingly impressive numbers close to the zero FDR of STADe with KNN having the best score of **5.08** e^{-6} while isolation forest and LOC scored **0.0196** and **0.0317** respectively. Such low FPR scores (i.e. below 0.04) could be misleading and would suggest that an anomaly detection model is recording relatively low false positives. However, a closer look at the confusion matrices in Fig. 10 reveals otherwise. For example, the isolation forest and LOC algorithms recorded 19,396 and 31,137 false positives respectively (shown in Figs. 10(b) and 10(c)) in the period under consideration (7.3 h) while KNN recorded only 5 (Fig. 10(a)) in the same period. The FDR metric reflects this performance more accurately with scores of **0.00033**, **0.3933**, and **0.6339** for KNN, isolation forest, and LOC respectively. This means that 39.33% of anomalies detected by the isolation forest algorithm within a 7.3 h period were false while for LOC and KNN it was 63.39% and 0.03% respectively. In reality, the FDR metric would be more beneficial than FPR in an operational environment. This is because when processing tens of millions of network data each day, even a modest false discovery rate can overwhelm a security analyst [32] - as can be seen with the high number of false positives recorded by the isolation forest and LOC algorithms within a 7.3 h operational period.

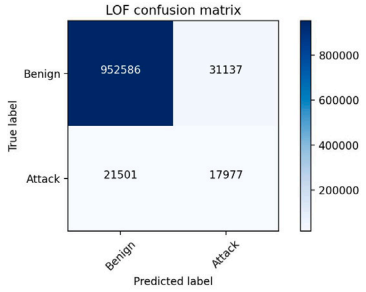
To summarise, the STADe methodology outperformed the KNN, isolation forest, and LOF algorithms in detecting anomalies in the industrial network dataset. However, with respect to FDR scores, the KNN algorithm performed closest to the STADe methodology. This may be because the KNN algorithm is a distance-based algorithm that ranks each point based on the distance between the point and the nearest point, and identifies the top-most points as anomalies. This is similar to the STADe methodology as they both calculate Euclidean distances between data points.



(a) KNN confusion matrix



(b) Isolation forest confusion matrix



(c) LOF confusion matrix

Fig. 10. Confusion matrices for KNN, isolation forest, and LOF.

Table 7
Evaluation metric scores for KNN, isolation forest, and LOC algorithms (for FPR/FDR lower is better).

Algorithm	Precision	Recall	F1-Score	FPR	FDR
KNN	0.999	0.379	0.55	5.08 e^{-6}	0.00033
Isolation Forest	0.607	0.755	0.673	0.0196	0.3933
LOF	0.366	0.455	0.408	0.0317	0.6339
STADe	**1.0**	**0.875**	**0.933**	**0.0**	**0.0**

7. Conclusions

Anomaly detection in industrial networks has had a problem with high false positive rates and high computational complexities which has hindered its widespread use in practice. The novel STADe methodology introduced in this paper represents an unsupervised time-window-based approach to anomaly detection in industrial control networks that results in zero false positives. This work aimed to explore a mechanism for detecting breaks in periodicity to flag anomalies. For this reason, a single feature of packet inter-arrival times was recorded as point events. The aim was to characterise the periodicity of any given industrial network using the packet timings to create a mini-model of the system representing the normal operation pattern. This mini-model is represented as the baseline window, which further acts as the sliding window that is used to compare with the rest of the traffic windows.

The results from the experiments showed no false positives with F1 scores of 0.97, 0.923, and 0.8 recorded for the detection of field flooding, SYN flooding, and MITM attacks respectively. In order to assess the performance of STADe in relation to other unsupervised machine learning algorithms, namely K-Nearest Neighbours (KNN), Isolation Forest, and Local Outlier Factor (LOF), a series of additional experiments were conducted. The results revealed F1 scores of 0.55, 0.673, and 0.408 for KNN, Isolation Forest, and LOF, respectively. Notably, STADe demonstrated superior performance, achieving an F1 score of 0.933 when applied to the same dataset. This makes STADe very promising to explore further.

Essentially, from the experiments, the STADe methodology was able to:

- measure the periodicity of a given industrial network in the form of a pattern,
- segment the network traffic into time windows which were further compared with each other to detect deviations from the normal pattern established in order to detect anomalies, and
- as an additional step, map this pattern onto a 3-dimensional space visually.

The fact that it utilises a single feature of packet timings that is unaffected by network encryption means it could potentially be integrated with other security solutions simultaneously to improve the security posture of industrial networks.

One important application in the real world for STADe is its potential usefulness if used in conjunction with a human-in-the-loop to narrow down large volumes of data and enable quick identification of anomalous packets within a time window and investigate further to determine the cause of the anomaly. For our specific test case in this study, with a window size of approximately 2500 packets, it essentially means that an attack can potentially be detected within a minute of it occurring. This could also potentially put an end to scenarios where attacks are carried out undetected for several months.

7.1. Limitations and future directions:

One of the limitations of this work is that this methodology would be more difficult to implement in an OT environment where there are intermittent control commands that may be part of the overall modus operandi of the plant. In other words, an OT environment that exhibits less periodicity than the norm. This may require a longer capture time to establish accurate baselines and thresholds for effective detection.

Additionally, as mentioned earlier, this study employed a static threshold to ensure simplicity and focus on validating the proof-of-concept. This approach provides a baseline for comparison, which can be essential for future studies aiming to build upon this work. However, future enhancements to this method could involve the use of a dynamic threshold that adapts to the statistical properties of the system's normal operation data. One possible approach would be to model the distribution of diff scores generated from normal traffic.

For instance, if the data approximately follows a Gaussian (normal) distribution, the threshold could be set at a specific number of standard deviations (e.g., 3σ) above the mean, corresponding to a desired false-positive rate. This would dynamically adjust the threshold to account for variations in the system's statistical characteristics, potentially improving the adaptability of the approach in dynamic environments. While dynamic thresholding presents an exciting avenue for future work, the static threshold used in this study effectively demonstrates the concept's viability and provides a strong foundation for further research.

Finally, during this study, there was a lack of diversity in datasets with different industrial protocols. The protocol used in all datasets used was ModbusTCP. Future work will focus on optimising the selection of window size and threshold hyper-parameters as they are the most sensitive to changes in the results. More industrial network pcaps utilising other widely used protocols (e.g. EthernetIP, DNP3, Common Industrial Protocol, OPC UA/DA) will also be explored to evaluate the effectiveness of the STADe methodology.

CRediT authorship contribution statement

Abubakar Sadiq Mohammed: Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Eirini Anthi:** Writing – review & editing, Supervision. **Omer Rana:** Writing – review & editing, Supervision. **Pete Burnap:** Supervision, Project administration, Funding acquisition. **Andrew Hood:** Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by the Petroleum Technology Development Fund, Nigeria (PTDF).

Data availability

Data will be made available on request.

References

[1] P. Schneider, K. Böttinger, High-performance unsupervised anomaly detection for cyber-physical system networks, in: Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy, 2018, pp. 1–12.
[2] D.-S. Kim, H. Tran-Dang, D.-S. Kim, H. Tran-Dang, An overview on industrial control networks, Ind. Sensors Control. Commun. Networks: From Wired Technol. To Cloud Comput. the Internet Things (2019) 3–16.
[3] K.O. Akpinar, I. Ozcelik, Methodology to determine the device-level periodicity for anomaly detection in ethercat-based industrial control network, IEEE Trans. Netw. Serv. Manag. 18 (2) (2020) 2308–2319.
[4] R.R.R. Barbosa, R. Sadre, A. Pras, Exploiting traffic periodicity in industrial control networks, Int. J. Crit. Infrastruct. Prot. 13 (2016) 52–62.
[5] L. Chen, S. Gao, B. Liu, An improved density peaks clustering algorithm based on grid screening and mutual neighborhood degree for network anomaly detection, Sci. Rep. 12 (1) (2022) 1409.
[6] S. Ali, S.U. Rehman, A. Imran, G. Adeem, Z. Iqbal, K.-I. Kim, Comparative evaluation of AI-based techniques for zero-day attacks detection, Electron. 11 (23) (2022) 3934.
[7] S. He, J. Zhu, P. He, M.R. Lyu, Experience report: System log analysis for anomaly detection, in: 2016 IEEE 27th International Symposium on Software Reliability Engineering, ISSRE, IEEE, 2016, pp. 207–218.
[8] M. Al-Qatf, Y. Lasheng, M. Al-Habib, K. Al-Sabahi, Deep learning approach combining sparse autoencoder with SVM for network intrusion detection, Ieee Access 6 (2018) 52843–52856.
[9] H. Hindy, D. Brosset, E. Bayne, A.K. Seeam, C. Tachtatzis, R. Atkinson, X. Bellekens, A taxonomy of network threats and the effect of current datasets on intrusion detection systems, IEEE Access 8 (2020) 104650–104675.
[10] K. Al Jallad, M. Aljnidi, M.S. Desouki, Anomaly detection optimization using big data and deep learning to reduce false-positive, J. Big Data 7 (1) (2020) 1–12.
[11] M. Zipperle, F. Gottwalt, E. Chang, T. Dillon, Provenance-based intrusion detection systems: A survey, ACM Comput. Surv. 55 (7) (2022) 1–36.
[12] S. Axelsson, The base-rate fallacy and the difficulty of intrusion detection, ACM Trans. Inf. Syst. Secur. (TISSEC) 3 (3) (2000) 186–205.
[13] J. Sadowski, Zero tolerance: More zero-days exploited in 2021 than ever before, 2022.
[14] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 1–58.
[15] C.-Y. Lin, S. Nadjm-Tehrani, M. Asplund, Timing-based anomaly detection in SCADA networks, in: Critical Information Infrastructures Security: 12th International Conference, CRITIS 2017, Lucca, Italy, October 8-13, 2017, Revised Selected Papers 12, Springer, 2018, pp. 48–59.
[16] D. Jiang, J. Liu, Z. Xu, W. Qin, Network traffic anomaly detection based on sliding window, in: 2011 International Conference on Electrical and Control Engineering, IEEE, 2011, pp. 4830–4833.
[17] A. Tekeoglu, K. Bekiroglu, C.-F. Chiang, S. Sengupta, Unsupervised time-series based anomaly detection in ICS/SCADA networks, in: 2021 International Symposium on Networks, Computers and Communications, ISNCC, IEEE, 2021, pp. 1–6.
[18] Y. Peng, C. Xiang, H. Gao, D. Chen, W. Ren, Industrial control system fingerprinting and anomaly detection, in: International Conference on Critical Infrastructure Protection, Springer, 2015, pp. 73–85.
[19] J. Van Splunder, Periodicity detection in network traffic, Tech. Rep. Math. Inst. Univ. Leiden (2015).
[20] J. Åkerberg, J. Furunäs Åkesson, J. Gade, M. Vahabi, M. Björkman, M. Lavassani, R. Nandkumar Gore, T. Lindh, X. Jiang, Future industrial networks in process automation: Goals, challenges, and future directions, Appl. Sci. 11 (8) (2021) 3345.
[21] L. Liang, J. Han-hong, R. Wan-zhi, W. Jie, Study on the characteristics of network traffic based on STFT, in: 2015 2nd International Conference on Information Science and Control Engineering, IEEE, 2015, pp. 485–488.
[22] C.V. Wright, F. Monrose, G.M. Masson, Using visual motifs to classify encrypted traffic, in: Proceedings of the 3rd International Workshop on Visualization for Computer Security, 2006, pp. 41–50.
[23] N. Hubballi, D. Goyal, Flowsummary: Summarizing network flows for communication periodicity detection, in: Pattern Recognition and Machine Intelligence: 5th International Conference, PReMI 2013, Kolkata, India, December 10-14, 2013. Proceedings 5, Springer, 2013, pp. 695–700.
[24] P. Indyk, N. Koudas, S. Muthukrishnan, Identifying representative trends in massive time series data sets using sketches, in: 26th International Conference on Very Large Data Bases, VLDB 2000, 2000, pp. 363–372.
[25] W. Mazurczyk, K. Szczypiorski, B. Jankowski, Towards steganography detection through network traffic visualisation, in: 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems, IEEE, 2012, pp. 947–954.
[26] A.S. Mohammed, E. Anthi, O. Rana, N. Saxena, P. Burnap, Detection and mitigation of field flooding attacks on oil and gas critical infrastructure communication, Comput. Secur. 124 (2023) 103007.
[27] A.S. Mohammed, P. Reinecke, P. Burnap, O. Rana, E. Anthi, Cybersecurity challenges in the offshore oil and gas industry: An industrial cyber-physical systems (ICPS) perspective, ACM Trans. Cyber- Phys. Syst. (2022) URL https://doi.org/10.1145/3548691.
[28] Y. Xiao, Analysis of deep anomaly detection algorithms, in: Proceedings of the 2021 6th International Conference on Multimedia Systems and Signal Processing, 2021, pp. 64–67.
[29] L. Ruff, J.R. Kauffmann, R.A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T.G. Dietterich, K.-R. Müller, A unifying review of deep and shallow anomaly detection, Proc. IEEE 109 (5) (2021) 756–795.
[30] K.P. Murphy, Machine Learning: a Probabilistic Perspective, MIT Press, 2012.
[31] A.J.J. Cespedes, B.H.B. Pangestu, A. Hanazawa, M. Cho, Performance evaluation of machine learning methods for anomaly detection in CubeSat solar panels, Appl. Sci. 12 (17) (2022) 8634.
[32] B. Anderson, D. McGrew, Identifying encrypted malware traffic with contextual flow data, in: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security, 2016, pp. 35–46.