Available online at www.sciencedirect.com

ScienceDirect

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

Computers & Security

An unsupervised anomaly-based detection approach for integrity attacks on SCADA systems

Abdulmohsen Almalawi [a,c], *Xinghuo Yu* [b], *Zahir Tari* [a], *Adil Fahad* [a,d,*], *Ibrahim Khalil* [a]

[a] School of Computer Science and Information Technology, RMIT University, Melbourne, Vic. 3001, Australia
[b] School of Electrical and Computer Engineering, RMIT University, Melbourne, Vic. 3001, Australia
[c] Faculty of Computing and IT King Abdulaziz University, Jeddah, Saudi Arabia
[d] Department of Computer Science, Al-Baha University, Al-Baha City, Saudi Arabia

ARTICLE INFO

Article history:
Received 7 November 2013
Received in revised form
25 May 2014
Accepted 13 July 2014
Available online 29 July 2014

Keywords:
Unsupervised detection
Cyber-warfare
SCADA systems
Intrusion Detection System
Consistent/Inconsistent SCADA Patterns

ABSTRACT

Supervisory Control and Data Acquisition (SCADA) systems are a core part of industrial systems, such as smart grid power and water distribution systems. In recent years, such systems become highly vulnerable to cyber attacks. The design of efficient and accurate data-driven anomaly detection models become an important topic of interest relating to the development of SCADA-specific Intrusion Detection Systems (IDSs) to counter cyber attacks. This paper proposes two novel techniques: (i) an automatic identification of consistent and inconsistent states of SCADA data for any given system, and (ii) an automatic extraction of proximity detection rules from identified states. During the identification phase, the density factor for the k-nearest neighbours of an observation is adapted to compute its inconsistency score. Then, an optimal inconsistency threshold is calculated to separate inconsistent from consistent observations. During the extraction phase, the well-known fixed-width clustering technique is extended to extract proximity-detection rules, which forms a small and most-representative data set for both inconsistent and consistent behaviours in the training data set. Extensive experiments were carried out both on real as well as simulated data sets, and we show that the proposed techniques provide significant accuracy and efficiency in detecting cyber attacks, compared to three well-known anomaly detection approaches.

1. Introduction

SCADA systems control and monitor industrial and infrastructure processes such as transportation, oil and gas refining and energy and water distribution networks (Yu et al., 2011; Fahad et al., 2013). In recent years, the incorporation of Commercial-Off-The-Shelf (COTS) products such as standard hardware and software platforms have begun to be used in SCADA systems. This incorporation allowed various products from different vendors to be integrated with each other to build a SCADA system at low cost. In addition, the integration of standard protocols (e.g. TCP/IP) into COTS products has increased their connectivity, thereby increasing productivity and profitability. However, this shift from proprietary and customized products to standard ones exposes these systems to cyber threats (Oman et al., 2000). Undoubtedly, any attack targeting SCADA systems could lead to high financial losses

* *Corresponding author.*
E-mail addresses: alharthi.adil@gmail.com, dr.alharthi.fahad@gmail.com (A. Fahad).
http://dx.doi.org/10.1016/j.cose.2014.07.005

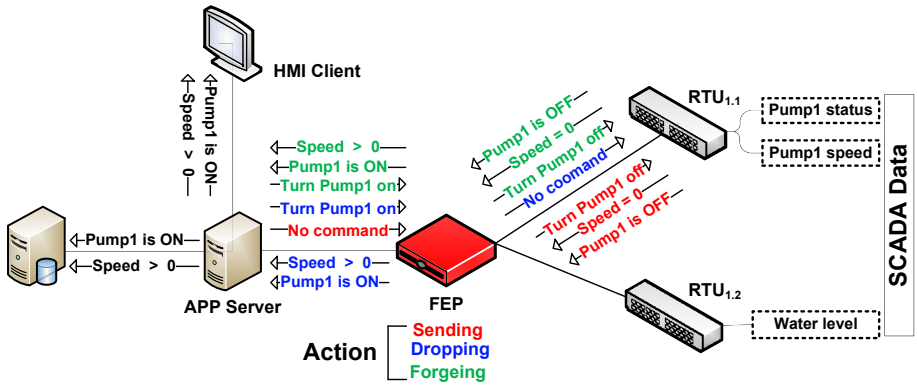

Fig. 1 – Compromised FEP sends undesired command and falsifies the feedback information.

and serious impacts on public safety and the environment. The attack on the sewage treatment system in Maroochy Shire (Australia) is an example of such attacks on critical infrastructures (Slay and Miller, 2007), where the attacker took over the control devices of a SCADA system. The Stuxnet (Falliere et al., 2011) worm, which was designed to damage nuclear power plants in Iran, is a recent example of threats targeting control systems. Both of the aforementioned attacks are classified as man-in-the-middle (MITM) attacks, where control devices are compromised to perform malicious actions, and meanwhile false information is sent to the Master Terminal Unit (MTU) to avoid detection. Such cyber threats allow attackers to perform high-level control actions (Wei et al., 2011; Queiroz et al., 2011; Nicholson et al., 2012), and pose potential threats to SCADA systems.

An awareness of the potential threats to, as well as the need to reduce the various vulnerabilities of SCADA systems have recently become an important research focus in the area of security. A number of (security) measures have been used in traditional IT systems, including management, filtering, encryption and intrusion detection. However, such measures cannot be directly applied to SCADA systems without considering their specific characteristics. Additionally, none of these traditional IT security solutions can completely protect SCADA systems from potential cyber attacks. However, properly adapting/extending such IT solutions can create robust protection of SCADA systems against cyber attacks. IDS (Intrusion Detection System) is one of the security solutions that has showed promising results in detecting malicious activities in traditional IT systems, and this is one of the reasons for using and adapting it to SCADA environments.

1.1. *Problem statement*

To illustrate the intrusion detection problem, two well-known scenarios (Verba and Milvich, 2008) are considered. Fig. 1 illustrates an attacker compromising the front end processor (FEP) by carrying out three actions: (i) initialising a connection with a remote terminal unit ($RTU_{1.1}$) and sending a command without receiving a corresponding command from the application server; (ii) dropping the command sent from the application server to $RTU_{1.1}$, and frogging feedback information sent back to the application server to meet the attack; and (iii) frogging the command sent from the application server to $RTU_{1.1}$, as well as frogging feedback information sent back from $RTU_{1.1}$ to the application server. All commands sent to $RTU_{1.1}$ will be trusted, as they are syntactically valid and sent from an FEP.

Two inconsistent data can be identified in this scenario: an **inconsistent network traffic pattern** and (ii) an *inconsistent SCADA data*. The former relates to the following: (i) an FEP is not an intelligent device that can make a decision and send a command to $RTU_{1.1}$ without receiving a corresponding command; (ii) and the dropped command at FEP will be shown up in the network stream from the application server to the FEP, but not in the network stream from the FEP to the $RTU_{1.1}$, while the frogged commands between the application server and $RTU_{1.1}$ can be identified by the inconsistent SCADA data.

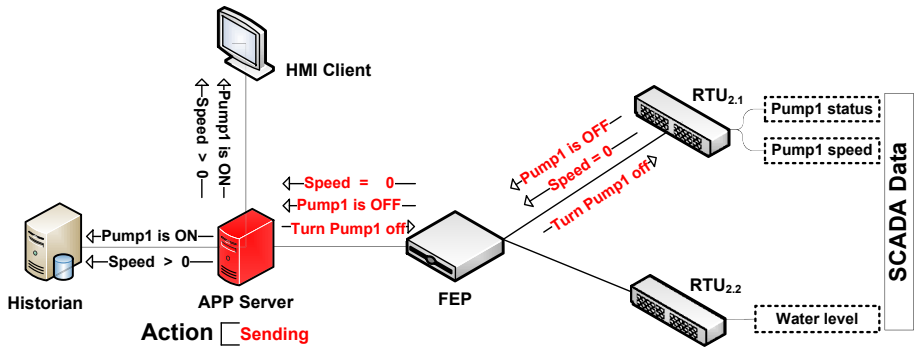

Fig. 2 – Compromised application server sending false information.

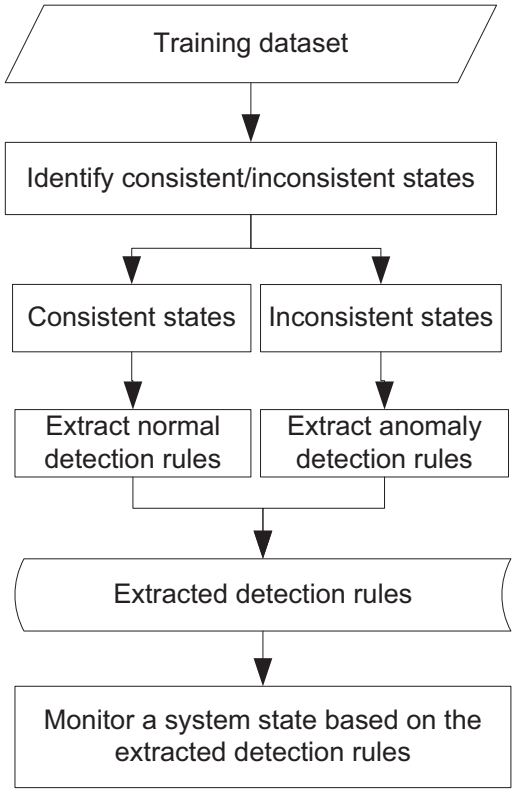

Fig. 3 – The steps of the proposed approach.

For example, the command in the network stream from the application server to the FEP shows that the status of $pump_1$ is ON, while in the network stream from the FEP to the $RTU_{1.1}$, it is OFF. Clearly, the inconsistencies in this scenario shows that the aforementioned MITM attacks are performed by the FEP. In what follow, however, we show a scenario where the monitoring of inconsistencies fails to detect MITM attacks.

Let's consider the example shown in Fig. 2. This example illustrates an attacker compromising an intelligent application server that can initiate independent actions. It drops commands sent from the operator, and therefore an unsafe situation could be created. An attacker initialises a command from the application server to turn off $pump_1$, and it can be seen that both the network traffic stream and the SCADA data between $RUT_{2.1}$ and the application server are consistent for this command. However, the SCADA data, such as the speed and the status of *pump*1, could be inconsistent with the sensory node of the water level in $RTU_{2.2}$, as they are set to values that violate the specifications of the system from the operational perspective.

The evolution of SCADA data can reflect the system's state: *consistent* or *inconsistent*. Therefore, the monitoring of the SCADA data has been proposed as an efficient tailored IDS for SCADA environments. The detection methods are broadly categorized into two types: signature-based and anomaly-based. The former can detect only an attack whose signature is already known, while the latter can detect unknown attacks by looking for activities that deviate from an expected patterns (or behaviours). Learning the anomaly-based detection models can be performed via three modes, namely *supervised*, *semi-supervised* and *unsupervised*. The class labels must be available for the first mode; however, this type of learning is costly and time-consuming because domain experts are required to label hundreds of thousands of data observations. The second mode is based on the assumption that the training data set represents only one behaviour, either normal or abnormal. There are a number of issues pertaining to this mode. The system has to operate for a long time under normal conditions in order to obtain purely normal data that comprehensively represent normal behaviours. However, there is no guarantee that any anomalous activity will occur during the data collection period. On the another hand, it is difficult to obtain a training data set that covers all possible anomalous behaviours that could occur in the future. Alternatively, the unsupervised mode can be an appropriate solution to address the aforementioned issues, where the anomaly detection models can be learned from unlabelled data without prior knowledge about normal/abnormal behaviours. However, the poor efficiency and low accuracy this type of learning are challenging.

1.2. *Contributions*

This paper proposes a novel unsupervised SCADA data-driven anomaly detection approach intended to be used as a passive SCADA IDS. That is, it only raises alarms when suspicious activities are detected, and the appropriate responses will be left for a system administrator. The SCADA data, which are generated by sensors/actuators, are used as valuable information in the proposed approach. Fig. 3 shows the two main steps of the proposed approach: the identification of consistent/inconsistent states from unlabelled SCADA data, and the extraction of proximity-based detection rules for each behaviour.

The use of control data has attracted the attention of many researchers studying SCADA data-driven anomaly detection models that are able to learn the mechanistic behaviour of SCADA systems without knowledge of the physical behaviour of such systems (Rrushi, April 2009; Marton et al., 2013; Gao et al., 2010; Zaher et al., 2009). Such studies however can operate only in two learning modes: *supervised* and *semi-supervised*. Despite the promising results of these learning modes, there are a number of issues that restrict their use (see the previous Section 1.1). This paper proposes an *unsupervised* learning approach, which consists of two novel techniques. The first one is used to identify consistent/inconsistent states from unlabelled data. This is performed by giving an inconsistency score to each observation using the density factor for the *k*-nearest neighbours of the observation. An optimal inconsistency threshold is later computed to separate inconsistent from consistent observations. The second proposed technique extracts proximity-based detection rules for each behaviour, whether inconsistent or consistent. During this phase, the fixed-width clustering technique (Eskin et al., 2002) is used to cluster each behaviour individually into micro-clusters with a constant fixed width, which is statistically determined. The centroids of all the created micro-clusters are used as the proximity-detection rules that are assumed to form a small and most representative data set for both inconsistent and consistent behaviours in the training data set.

The proposed approach is evaluated on both real and simulated data sets; two are generated by a simulation of a SCADA system that uses well-known models as discussed in Section 4.1, while the third is real and consists of consistent/inconsistent observations. In particular, we compared the effectiveness of our unsupervised approach with existing unsupervised and semi-supervised anomaly detection approaches.

1.3. *Organization of the paper*

This paper is organised as follows. Section 3 provides a characterisation of consistent/inconsistent observation states for SCADA data, as well as the details of the proposed approach. Section 4 presents the experimental setup, followed by results and analysis in Section 5. Finally, we conclude the work in Section 6.

2. Related work

In the design of an IDS, two main processes are often considered. First is the selection of the information source (e.g. network-based, application-based) to be used, through which anomalies can be detected. Second is the development of a learning (or analysis) method that is used to efficiently build the detection model using the specified information source. SCADA-specific IDSs can be broadly grouped into three categories in terms of the latter process: misuse (signature-based) detection (Digitalbond, 2013), anomaly detection (Linda et al., 2009; Kumar et al., 2007; Valdes and Cheung, 2009; Yang et al., 2006; Ning et al., 2002; Gross et al., 2004) specification-based detection (Cheung et al., 2007; Carcano et al., 2011; Fovino et al., 2010; Fernandez and Larrondo-Petrie, 2010; Alcaraz and Lopez, 2014a, 2014b). Recently, several other signature-based rules (Digitalbond, 2013) have been proposed to specifically detect particular attacks on SCADA protocols. These rules can perfectly detect *known attacks* at the SCADA level.

To detect *unknown attacks* at the SCADA network level, a number of approaches have been proposed. O. Linda et al. (Linda et al., 2009) suggested a window-based feature extraction technique to extract important features of SCADA network traffic, and then used a feed-forward neural network with a back propagation training algorithm for modelling the boundaries of normal behaviours. This approach requires, however, substantial execution time during the training phase, in addition to the need for re-learning the boundaries of normal behaviours upon receiving new behaviours. The model-based detection approach proposed by A. Valdes and S. Cheung (Valdes and Cheung, 2009) learns about the communication patterns. This is based on the claim that such patterns are regular and predictable because SCADA has specific services as well as interconnected and communicating devices that are already predefined. This approach is useful in providing a border monitoring of the requested services and devices. Similarly, P. Gross et al. (Gross et al., 2004) proposed a collaborative approach, named *selecticast*, which uses a centralised server to dispatch among ID sensors information about activities coming from suspicious IPs. In essence, all these approaches fail to detect **high-level control attacks**, which are the most difficult threats to defend against (Wei et al., 2011). Furthermore, SCADA network-level approaches are not concerned with the operational meaning of the SCADA data, which are carried by SCADA protocols, as long as they are not violating the specifications of the protocol being used, or a broader picture of the monitored system.

Thus, analytical models based on full system specifications have been proposed in the literature. Fovino et al. (Fovino et al., 2010) came up with an analytical approach to identify critical states for specific-correlated process parameters. Such a detection model is used to detect malicious actions (e.g. high-level control attacks) that transform the targeted system into a critical state. In the same direction, Fovino et al. (Fovino et al., 2012) and Carcano et al. (Carcano et al., 2011) extended this idea by identifying critical states for specific-correlated process parameters. Each critical state is represented by a multivariate vector, where a vector being a reference point to measure the degree of criticality of a system. For example, when the distance of a system's state is close to any critical state, it shows that it is approaching a critical state. However, critical state-based approaches require full specifications of all correlated process parameters, and this in addition to their respective acceptable values. Moreover, the analytical identification of critical states for a large number of correlated process parameters is time-expensive and difficult. This is because the complexity of the inter-relationships amongst parameters is proportional to their number. Additionally, any change made by either adding or removing process parameters would require making the same effort again. Obviously, human errors are highly expected in the identification process of critical system states.

Due to the aforementioned issues relating to analytical models, SCADA data-driven models have been proposed to capture the mechanistic behaviour of SCADA systems without any knowledge of their physical behaviour. It was experimentally shown by Wenxian and Jiesheng (Wenxian and Jiesheng, 2011) that operational SCADA data for wind turbine systems are useful if they are properly analysed to indicate the condition of the system that is being supervised. A number of SCADA data-driven methods for anomaly detection have been proposed in the literature. Jin et al. (Jin et al., 2006) extended the set of invariant models using *a value range model* to detect anomalous data in the values for a particular process parameter. A pre-determined threshold is proposed for each parameter, and any value exceeding this threshold is considered as *anomalous*. This approach can detect anomalous values of individual parameters, while it fails to detect values whose abnormality is caused by a certain value of another parameter. Such types of parameters are called *multivariate parameters*, and are assumed to be directly (or indirectly) correlated. Rrushi et al. (Rrushi, April 2009) applied probabilistic models to estimate the normalcy of the evolution of values of multivariate process parameters. Similarly, Marton et al. (Marton et al., 2013) proposed a data-driven approach to detect abnormal behaviour in industrial equipments, where two multivariate analysis techniques, namely principal component analysis (PCA) and partial least squares (PLS), are combined to design the detection models. Neural network-based approaches have also been proposed to model the

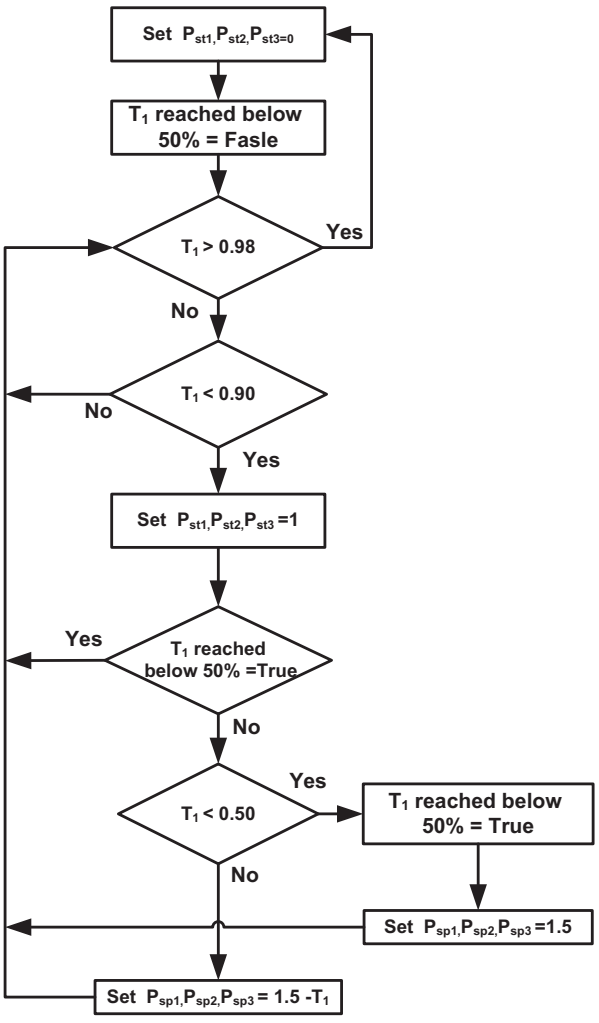

Fig. 4 – The normal operation of the physical nodes P_{st_1}, P_{st_2}, P_{st_3}, P_{sp_1}, P_{sp_2}, P_{sp_3}.

normal behaviour of various SCADA systems. For example, Gao et al. (Gao et al., 2010) proposed a neural network-based intrusion detection system for a water tank control system, which was later extended by Zaher et al. (Zaher et al., 2009) to build model of the normal behaviour for a wind turbine to identify faults (anomalies).

3. The proposed intrusion detection approach

This section describes consistent/inconsistent states of SCADA data, as well as the techniques that contribute to the development of an unsupervised intrusion detection method to detect SCADA-based integrity attacks. Specifically, the proposed approach consists of (i) a technique that identifies consistent and inconsistent multivariate SCADA data, and (ii) a technique that extracts proximity-based detection rules used to perform a near-real-time monitoring of integrity attacks. Fig. 3 illustrates the different steps of the proposed approach. This approach is based on density-based outlier detection during the identification phase and a fixed-width clustering technique for extracting detection rules. The time complexity related to data dimensionality is not an issue, as the proposed approach performs both phases in an off-line mode. Our approach also uses the concept of distance ranking instead of absolute distance (Angiulli and Pizzuti, 2005) during the identification of consistent/inconsistent SCADA data.

3.1. *A state of SCADA data*

SCADA data, such as sensor measurements and actuator control data, are data sources for the proposed IDS. The consistency of such data represents the normal system state, while any inconsistency indicates a malicious action. Consistent data are defined by the specifications which describe the valid/acceptable data in terms of the system's operational perspective. From the simulation presented in Section 4.1, Fig. 4 depicts the (normal) operation producing consistent SCADA data observations of the physical nodes P_{st_1}, P_{st_2}, P_{st_3}, P_{sp_1}, P_{sp_2}, P_{sp_3} and T_1, where P_{st_i} and P_{sp_i} represent the status and the speed of $Pump_i$, and T_j represents the water level in the $Tank_j$.

Definition 1. (State). *A state represents a combination of SCADA data produced by nodes at a certain period of time t. A state for n nodes can be represented by a vector $p \in R^n$. States can be finite if the values of the nodes are discrete, or infinite when at least one of the node data is continuous.*

Definition 2. (Consistent/Inconsistent state). *A consistent state is a state that statistically has the higher likelihood of being generated by the same mechanism that generated the majority of states. An inconsistent state is any state that statistically deviates from the majority of the states.*

3.2. *Identification of consistent/inconsistent states*

The identification step, as depicted in Fig. 3, is the first phase in separating inconsistent SCADA data observations from the consistent ones. To perform this with unlabelled data, two assumptions are made: (i) the number of consistent SCADA data observations vastly outperform the inconsistent ones, and (ii) the inconsistent SCADA data observations must be statistically different from the consistent ones. Therefore, the proposed approach would be inappropriate for any situation that does not satisfy the two mentioned assumptions. The preliminary investigations show that inconsistent SCADA data observations have a similar definition of outliers in *n*-dimensional space, and are sparsely distributed in an informal way. That is, they could take various densities of *n*-dimensional space. Two steps are involved in identifying consistent/inconsistent SCADA data observations, as detailed below.

3.2.1. *Inconsistency scoring*

We propose an inconsistency scoring technique using a hybrid of local and global outlier detection approaches (Breunig et al., 2000, 1999; Vydunas, 2004; Arning et al., 1996). In the local outlier detection approach (Breunig et al., 2000,

1999), only local outliers are detected. This works well in a particular application domain whose normal behaviour forms a number of clusters that have different densities. In particular, the global approach (Vydunas, 2004; Arning et al., 1996) does not work well when the outliers are contained in the reference points. Since the initial investigation revealed different densities in inconsistent SCADA data observations, neither local nor global approaches are appropriate solutions for our problem. The choice of the best approach should not be predominantly influenced by either local or global approaches. Therefore, the proposed inconsistency scoring technique will need to rely on an average of similarity of the nearest neighbours and the number of neighbours k that play a major role in the influence of local and global approaches. The larger the value of k is, the stronger the influence of the global approach. For example, when k equals the size of the data set DS, the inconsistency score for SCADA data observation s_i is the average similarity from s_i to all observations in DS. This is similar to the global approaches proposed in Vydunas (2004); Arning et al. (1996).

Let D_{points} be monitored SCADA data observations, where $D_{points} = \{d_1,\cdots,d_n\}$. Let X be a vector of values of d_i, in the form of $X = \{x_1,x_1,\cdots,x_m\}$. Let S_i denotes an observation of D_{points}, $S_i = \{s_{i,1},s_{i,2},\cdots,s_{i,j}\}$, $i = 1,\cdots,m$, $j = 1,\cdots,n$ where m is the number of observations, and n the number of nodes. Any observation S_i can be either consistent or inconsistent. Given a data set of m observations, $DS = \{s_1,s_2,\cdots,s_m\}$. In the identification phase, a set of consistent observations, say $R = \{r_1,r_2,\cdots,r_k\}$, and a set of inconsistent observations, say $O = \{o_{k+1},o_{k+2},\cdots,o_m\}$, are identified, where $R \subseteq DS$, $O \subseteq DS$, $R \cup O = DS$, $R \cap O = \phi$.

To compute the inconsistency score for each observation S_i, which is expected to consist of hundreds of dimensions, we use the cosine similarity metric, which can work with sparse numeric data and high dimension space. This metric is used to measure the similarity between two vectors of n-dimensions, and it is widely used in document clustering and information retrieval (Steinbach et al., 2000). Let Ω be the function of cosine similarity between two observations $s = (s_1,\ldots,s_n)$ and $p = (p_1,\ldots,p_n)$, defined as follows:

$$\Omega(s,p) = \cos(\theta) = \frac{\sum_{i=1}^{n}(s_i \times p_i)}{\sqrt{\sum_{i=1}^{n}(s_i)^2} \times \sqrt{\sum_{i=1}^{n}(p_i)^2}} \quad (1)$$

Let k be a positive parameter such that $2 \le k \le |DS|$, and Ψ and Υ be the functions of an inconsistency score and the k-nearest neighbours for the observation s_i respectively. The inconsistency score for an observation is defined as follows:

$$\Psi(s_i,DS,k) = \frac{1}{k}\sum_{i=0}^{k}\Omega(s_i,\Upsilon(s_i,DS/s_i,k)) \quad (2)$$

Let us consider Fig. 5, where a data set D of eleven observations of $\mathbb{R}^2$ is shown. Let $k = 4$. In this example, the inconsistency score for the observation O_1, based on the Euclidean distance, is computed as follows. We first find the 4-nearest neighbours of O_1 using the function Υ

$$\Upsilon(O_1,D/O_1,4) = \{s_3,s_5,s_6,s_7\}$$

Then, the inconsistency score for the observation O_1 is computed as the average distance from O_1 to $\{s_3, s_5, s_6, s_7\}$.

$$\Psi(O_1,D,4) = \frac{27.7+22.4+24.6+20.3}{4} = \frac{95}{4} = 79.775$$

where, the numerators represent the Euclidean distances of the 4-nearest neighbours of O_1.

Fig. 5 shows the k-nearest neighbours of the observation s_i. For example, the k-nearest neighbours of the observation 1, which is a consistent observation with respect to $k = 5$, are $2,\cdots, 5$. This represents a cluster c_1 whose centre and size are observations 1 and k respectively. The inconsistency score of observation 1 is measured by the average distance of all observations $2,\cdots, 5$ in c_1 to observation 1. It can be seen that clusters c_1, c_2 and c_3 in Fig. 5 have similar radii, and this suggests that their centroid observations may have a similar inconsistency score. In fact, this is not always true because the inconsistency score is computed by the inter-compactness of the cluster, and not by reachability-distance (Ankerst et al., 1999). Since observations 2 and 5 are the centroids of clusters whose radii are larger than c_1, c_2 and c_3, they will obtain higher inconsistency scores. The radii of clusters c_6 and c_7, whose centroids are given by the inconsistent observations O_1 and O_2 respectively, are clearly the largest radii, and are considered as the inconsistent observations because they have relatively large radii that can significantly deviate from the mean of the radii of other observations. Algorithm 1 summarises the calculation steps of inconsistency scores for each observation s_i.

Algorithm 1 Inconsistency scoring algorithm

```
Input
DS: unlabelled SCADA data;
k: # of nearest neighbors;
Output
IncList        %list of inconsistency scores
for i := 1 To |DS| do
    IncList[i] = Ψ(DS[i], DS)
end for
```

3.2.2. Separation cut-off score

Since all observations are assigned with inconsistency scores, an appropriate cut-off score is required to determine whether the observation is consistent or inconsistent. Clearly, labelling consistent observations as inconsistent will result in a high false positive rate. Based on our assumption that consistent observations constitute a large portion of the training data set, they will also have very similar inconsistency scores. On the other hand, inconsistent observations are assumed to constitute a tiny portion of all observations, with high inconsistency scores, which are also assumed to be greater than the inconsistency scores of consistent observations. Under these assumptions, and in addition to the experimental results shown in Fig. 6, the probability density of inconsistency scores for observations is approximately normal and skewed to the right. Moreover, we assume that the inconsistency scores of a large enough number of observations (greater than 30) satisfy the Central Limit theorem (Johnson and Wichern, 2002) and can be approximately normally distributed. Since inconsistent observations are assumed to have significant inconsistency scores, they are considered to be beyond $\mu + 3\sigma$.

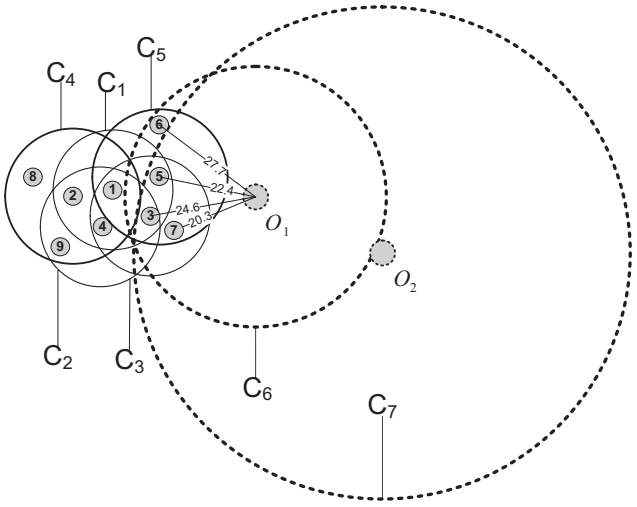

Fig. 5 – Illustration of inconsistency score based on intra-cluster cohesion factor.

Where, μ and σ are the mean and standard deviation of inconsistency scores for all observations in the training data set *DS* respectively, and they are defined as follows:

$$\mu = \frac{1}{|DS|}\sum_{s_i \in DS} \Psi(s_i, DS/s_i, k) \tag{3}$$

$$\sigma = \frac{1}{|DS|}\sqrt{\sum_{s_i \in DS} (\Psi(s_i, DS/s_i, k) - \mu)^2} \tag{4}$$

Therefore, the status of any observation s_i in the training data *DS* set is determined as follows:

$$Status(s_i) = \begin{cases} \Psi(s_i, DS, k) \le \mu + 3\sigma & \text{consistent} \\ Otherwise & \text{inconsistent} \end{cases} \tag{5}$$

Here, $\Psi(s_i, DS, k)$ is defined by Equation (2), which computes the inconsistency score for the observation s_i. See the illustrative example in Section 3.2.1 that shows how the score is computed.

3.3. *Extracting proximity-detection rules*

The various constraints of SCADA systems (e.g. real-time nature, lack of memory resources, limited computation power) require a tailored IDS that monitors a target system in at least near real-time and operates in a resource-constrained environment. In practice, feeding an IDS with the identified consistent and inconsistent observations for monitoring purposes is not a practical approach because (i) a large memory capacity is needed to store all learned observations, and (ii) it is time-consuming to calculate the similarity between the current observation and each learned observation during the monitoring phase. In this section, a detection rule extraction technique is proposed to extract a few detection rules which fully represent the entire identified observations. From these detection rules, the essential requirements of the SCADA-based IDS (e.g. a small memory capacity and low computation time) are met.

As shown in Fig. 3, detection rule extraction comes after the identification phase of consistent and inconsistent observations. The set of consistent observations is denoted as $R = \{r_1, r_2, \cdots, r_k\}$ and the set of critical states is denoted as $O = \{o_{k+1}, o_{k+2}, \cdots, o_m\}$, where $R \cap O = \phi$. It is assumed that the consistent observations of a number of nodes will form dense areas and will constitute a large portion of a training data set, while the inconsistent observations will be sparsely distributed in the *n*-dimensional space and constitute a tiny portion. Hence, *R* has one or more high density clusters. As we are interested mainly in extracting a few detection rules which can represent the learned model, we adopted the fixed-width clustering technique (Eskin et al., 2002) to cluster the consistent and inconsistent observations into micro-clusters with a constant fixed-width. However, choosing the appropriate fixed-width value is a challenging task. This is because a large width will degrade the model accuracy, while a small width will result in many rules that need to be checked in the detection phase. Since an inconsistency score for each observation is the average area of the neighbourhood of that observation, the mean of inconsistency scores is proposed as an appropriate value of the fixed-width parameter *w*, and this is defined as follows:

Let *X* be a vector of inconsistency scores of the training data set D_{points}, $X = \{x_1, x_2, \ldots x_n\}$, where *n* is the number of observations. Then:

$$w = \frac{1}{n}\sum_{i=1}^{n} X(i) \tag{6}$$

Algorithm 2 summarises the steps that are necessary to extract proximity-based detection rules for both consistent and inconsistent behaviour.

Algorithm 2 Proximity-based detection rules algorithm

Input
w: width of a cluster as Eq 6;
Data; %List of observations for one behaviour, whether consistent or inconsistent;
Initialize the set of cluster $\xi \longleftarrow \emptyset$;
Initialize the number of clusters $M \longleftarrow 0$;
for training sample $c_i \in Data$ **do**
  if $M == 0$ **then**
    Make a new cluster χ_1 with centroid $\chi_1^{\setminus}$ from c_i
    $\chi_1 \leftarrow \{c_i\}$; $\chi_1^{\setminus} \leftarrow c_i$;
    $\xi \leftarrow \{\chi_1\}$; $M \leftarrow M + 1$;
  else
    Find the most similar cluster χ_n to c_i
    $n = \max_{i \in k}\ \Omega(c_i, \chi_n)$ % where $k = 1 \ldots M$ and Ω is the cosine similarity
    if the similarity $n < w$ **then**
      Add c_i to cluster χ_n and update cluster centroid $\chi_n^{\setminus}$
      $\chi_n \leftarrow \chi_n \cup \{c_i\}$;
    else
      $M \leftarrow M + 1$;
      Make a new cluster χ_M with centroid $\chi_M^{\setminus}$ from c_i
      $\chi_M \leftarrow \{c_i\}$; $\chi_M^{\setminus} \leftarrow c_i$;
      $\xi \leftarrow \xi \cup \{\chi_M\}$;
    end if
  end if
end for
return all Clusters' centroids $\chi^{\setminus}$ % The returned centroids will be used as proximity detection rules

3.4. *Inconsistency detection*

The proximity-based detection rules, whereby each rule is represented by a cluster's centroid, are used to monitor any

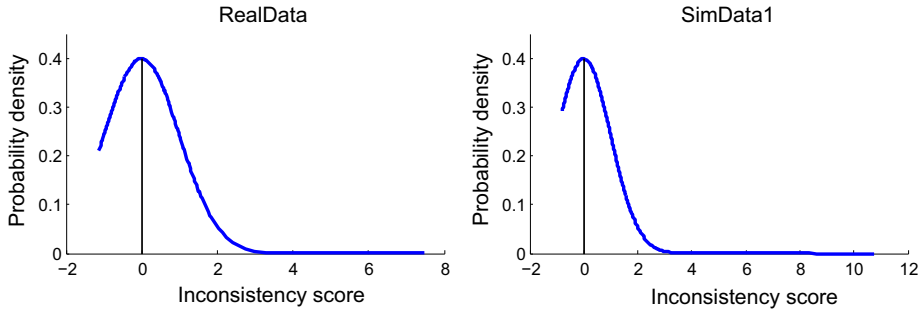

Fig. 6 – Standard probability density distribution of inconsistency scores for each observation in RealData and SimData1 are approximately normal and skewed to right.

observation for the target system to asses whether the current observation is consistent or inconsistent. Therefore, a current observation is labelled with the label of the closest micro-cluster. Let s_i be the current observation, $C = \{c_1, c_2, \ldots, c_n\}$ be the centroids of consistent micro-clusters and $O = \{o_1, o_2, \ldots, o_n\}$ be the centroids of inconsistent clusters. The closest consistent and inconsistent micro-clusters to s_i are determined by the following equations respectively:

$$c_{max} = \max_{c \in C} \Omega(s_i, c) \tag{7}$$

$$o_{max} = \max_{o \in O} \Omega(s_i, o) \tag{8}$$

where Ω is Equation (1), which computes the cosine similarity between two observations. In this case, the cosine similarity between the current (testing) observation s_i and all micro-clusters that represent the consistent behaviours, are computed. Then, the maximum cosine similarity degree is set to the variable c_{max}. Similarly, the computation of cosine similarity between s_i and all micro-clusters that represent the inconsistent behaviours is performed, and the maximum cosine similarity degree is set to the variable o_{max}. Afterwards, as defined in the following, the status of the current observation s_i is judged as *consistent* when the cosine similarity degree c_{max} is greater than o_{max}. Otherwise, it is judged as *inconsistent*.

$$StateType(s_i) = \begin{cases} c_{min} > o_{min} & \text{Consistent} \\ Otherwise & \text{Inconsistent} \end{cases} \tag{9}$$

3.5. Complexity analysis

We analyse the time complexity for each phase of the proposed approach, namely, (i) the calculation of the inconsistency score for each observation based on its neighbourhood density, (ii) the extraction of detection rules and (iii) the inconsistency detection phase. As previously mentioned, both first and second phases are assumed to be preformed in an off-line mode.

During the inconsistency scoring phase, k-nearest neighbours for each observation are required to be located in order to calculate the inconsistency factor based on neighbourhood density. To find the k-nearest neighbours for an observation, all observations in the training data set have to be checked. Let n be the number of observations in the training data set. The computational time for this process is O (n^2). Obviously, this process is expensive, especially with a large training data set. However, it is efficient compared to expert's involvement for labelling observations. Moveover, the computational time can be significantly reduced when tree-based spatial algorithms such as KD-tree (Sproull, 1991), Ball tree (Fukunaga and Narendra, 1975) and Cover tree (Beygelzimer et al., 2006) are implemented, where the observations are structured in a way that can efficiently accelerate the search for k-nearest neighbours. For instance, the use of Ball tree algorithm can efficiently reduce processing time to O ($n \log n$).

The second phase uses the fixed-width clustering approach to extract proximity-detection rules. A single pass is required over the data to individually partition the observations for each behaviour (whether inconsistent or consistent) into micro-clusters with constant-width w. The cosine similarity between each observation belonging to a particular behaviour, and all existing micro-clusters which are created from that behaviour, is computed. Hence, the computational complexity for partitioning each behaviour is O (mc), where m is the number of observations for that behaviour, while c ($\ll m$) is the number of the micro-clusters created from that behaviour as well.

Finally, we analyse the computational and memory complexity of the inconsistency detection phase. Since the centroids of micro-clusters of both inconsistent and consistent behaviours are only used as proximity detection rules, the computational and memory complexity of these rules is O ($C_n + O_c$), where C_n and O_n are the number of micro-clusters of inconsistent and consistent behaviours respectively. For the inconsistency detection process, the proposed approach does not need to keep all data in memory, but it only keeps the centroids of the micro-clusters that will be scanned in one pass for each testing observation, and therefore the detection phase is linear with the size of the proximity-detection rules.

4. Experimental setup

The main focus of this section is to set up an experimental environment to evaluate the robustness of the proposed approach. In what follows, we describe the simulation system used and two integrity attacks. We also describe the data sets

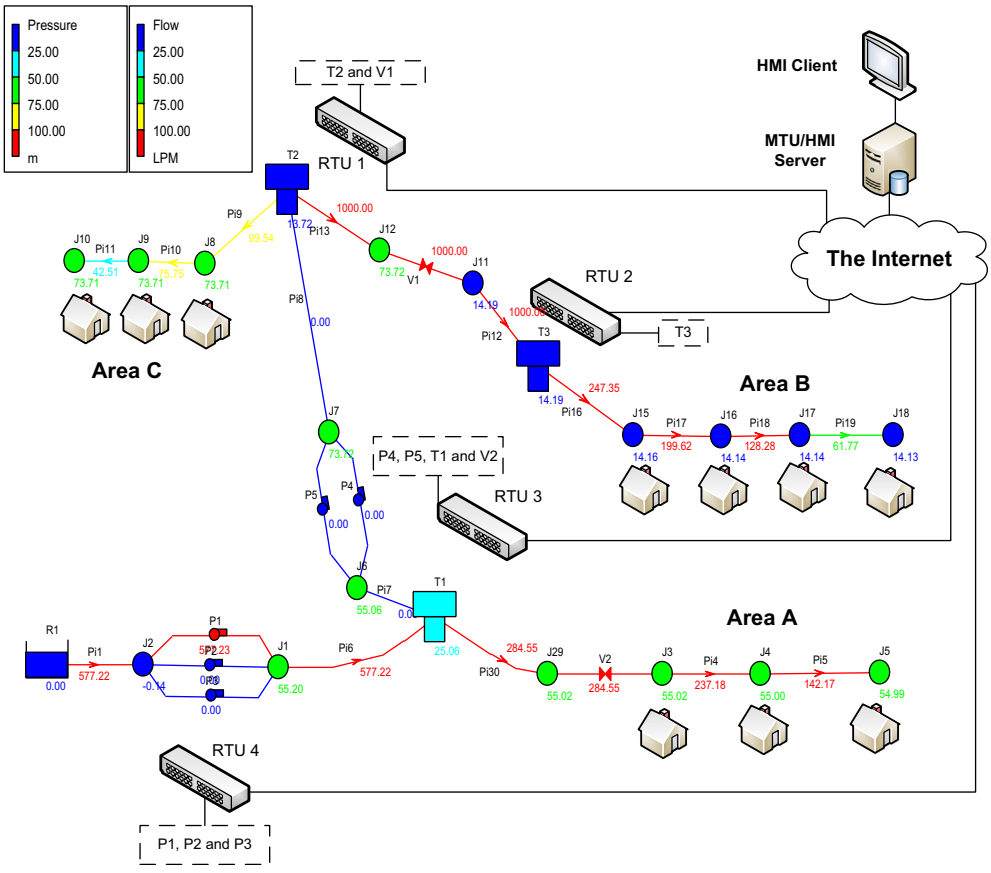

Fig. 7 – Simulation of a water distribution system.

used and the experimental parameters chosen for this evaluation.

4.1. *Simulation system setup*

As the proposed detection approach is intended to monitor control systems using SCADA data, a supervised infrastructure (e.g. power energy grids or water supply network) needs to be built to evaluate the proposed approach. However, building a real SCADA lab is too expensive in terms of cost, time and space, especially when supervised infrastructures need to be analysed and assisted against SCADA-based attacks. Therefore, we opted to implement a full virtual SCADA lab and simulate a water distribution system (WDS) as the supervised infrastructure. In practice, we used visualization features to represent the key parts of the SCADA system. For example, the field remote terminal units (RTU), Master Terminal Unit (MTU) and human–machine interface (HMI) are represented by a number of virtual machines after installing the library (Team, 2012) of the widely used Modbus protocol (IDA, 2013). The virtualized network is used as the communication infrastructure at all SCADA network levels (e.g. field and control levels).

To simulate a supervised infrastructure, we used the library of the well-known and free hydraulic and water quality model, called EPANET (Lewis, 2011). A WDS server acts as a surrogate for a real WDS. The EPANET model involves three modules, namely hydraulic, water quality and water consumption modules. We fed the consumption module with a specific model (i.e. the 2010 Melbourne water consumption (Melbourne Water, 2009)) so as to simulate the realistic behaviour of a water distribution system. One virtual machine is assigned to the WDS server. This server feeds the simulated data to virtualized field devices, and receives the Modbus/TCP control messages via a proxy. This proxy is used as an interface between virtualized field devices and the WDS server. For realistic simulation, the WDS server reads and controls process parameters such as water flow, pressure and valve status in response to message commands from a field device. The manipulated process parameters in the WDS server are:

- Water flow, pressure, demand and level.
- Valve status and setting.
- Pump status and speed.

4.2. *A Water Distribution System (WDS) scenario*

Fig. 7 depicts an example of a simple WDS for a small town. This town is divided into three areas, namely A, B and C. Each area has an elevated tank to supply it with water at a satisfactory pressure level. The supplied water is pumped out by three pumps from the treatment system into $Tank_1$. The water is also delivered to $Tank_2$ by two pumps. $Tank_3$ is supplied through gravity because of the elevation of $Tank_2$ which is higher than $Tank_3$. $Tank_1$ is twice as big as $Tank_2$ and $Tank_3$

because it is considered to be the main water source for areas B and C.

The water network is monitored and controlled by the SCADA system. In this scenario, four *RTUs*, namely RTU_1, …, RTU_4, are used. The *MUT* server plays a key role in sending command messages, in addition to storing the acquired data in the Historian. The *MUT* server sends command messages to the *RTUs* to functionally perform the following tasks:

- RTU_1 opens and closes valve V_1 according to the water level reading of $Tank_3$, and reads the water level of $Tank_2$.
- RTU_2 reads the water level of $Tank_3$.
- RTU_3 controls the status and speed of the pumps P_4 and P_5 according to the water level reading of $Tank_2$, and to read the water level of $Tank_1$.
- RTU_4 controls status and speed of the pumps P_1, P_2 and P_3 according to the water level reading of $Tank_1$.

4.3. Scenario of attacks

The purpose of this scenario is to affect the normal behaviour of WDS. The public network (e.g. Internet) is used to interconnect all the WDS's components through Ethernet modules. The Modbus/TCP application protocol is set up as a communication protocol. However, all TCP vulnerabilities are inherited and therefore the system is susceptible to external attacks such as DoS (Denial of Service) and integrity attacks. Refer to previous work (Queiroz et al., 2011; Almalawi et al., 2013; Fahad et al., 2014) for such attack scenarios on SCADA systems.

We have opted to simulate *man-in-the-middle* attacks. Such attacks require a prior knowledge of the target system, and this can be obtained from the specifications, or by correlation analysis for the network traffic of that system. By looking at the specifications, RTU_4 controls the status and the speed of pumps P_1, P_2 and P_3 in accordance with the water level reading of $Tank_1$. This is automatically performed by the MUT server. We compromised the MUT server and performed two malicious actions. Firstly, we sent 100 successive command messages to RTU_4 to turn off $pump_2$ and $pump_3$, and randomly change the speed of the $pump_1$ between 0.1 and 1. Secondly, we performed a similar malicious action on RTU_4 to turn off $pump_2$ and randomly change speed of $pump_1$ and $pump_2$ between 0.1 and 1. These types of attacks can be launched in a number of ways, and RTU_4 will trust such commands, as they are sent from the MUT server.

4.4. The data sets

To provide quantitative results for the proposed approach, we evaluated its performance using three different data sets: one is a publicly available, real data set (Frank and Asuncion, 2013) and two are generated by the aforementioned simulation system. The data sets contain raw records of SCADA data with an associated label to indicate whether the record was part of a consistent or inconsistent observation.

The simulated data sets consist of twenty-three nodes. Each data set consists of approximately 10,500 observations. The first data set has 100 inconsistent observations since $pump_2$ and $pump_3$ were turned off, although the water level in $tank_1$ was low; moreover, the speed of $pump_1$ was not properly adjusted according to the water level in $Tank_1$. Similarly, the second data set also had 100 inconsistent observations, since $pump_3$ was turned off although the water level in $Tank_1$ required this pump to be ON; in addition, the speeds of $pump_1$ and $pump_2$ were not properly adjusted according to the water level in $Tank_1$. The simulated data sets are denoted as SimData1 and SimData2 .

The real data set comes from the daily measuring of sensors in an urban waste water treatment plant (referred to as DUWWTP), and it consists of 38 data nodes (Frank and Asuncion, 2013). This data set consists of approximately 527 observations, 14 of which are labelled as inconsistent.

To improve the accuracy and efficiency of the proposed approach, the normalization technique is applied to all testing data sets to scale features in the range between 0.0 and 1.0. This prevents those features with a large scale from outweighing features with a small scale. As the actual minimum/maximum of features are already known, and because the identification process is performed in static mode, the min−max normalization technique is used to map the values of features. A given feature A would have a value in the range [0.0, 1.0]. Let us denote by min_A and max_A the minimum and maximum value of A respectively. Then, to produce the normalized value of v ($v \in A$) using the min−max normalization method, which we denote as v', the following formula is used:

$$v' = \frac{v - min_A}{max_A - min_A} \tag{10}$$

One parameter is required for the proposed approach. The k-nearest neighbours parameter is the influencing factor for the inconsistency scoring technique. However, this value could be application-dependent. Therefore, the value of k can be heuristically determined by setting this value at 5% of the representative data set, as it is assumed to be the maximum percentage of anomalies in a data set (Eskin et al., 2002).

5. Results and analysis

This section evaluates the accuracy of anomaly detection of the proposed unsupervised approach, and in addition, a comparison between this approach and two existing *unsupervised* and *semi-supervised* anomaly detection approaches is carried out. The detection accuracy for each approach is separately evaluated because the existing approaches that have been chosen as a basis for comparison with the proposed approach are inherently different in terms of the required parameters for learning anomaly detection models.

All observations in the three data sets are used to demonstrate the accuracy of the identification process for consistent/inconsistent observations in each data set. To evaluate the robustness of each approach, the k-fold-cross validation technique is applied to each data set for each experimental parameter. This technique divides the data set into K equal size subsets. Each time, one of the subsets is used as the validation data to test the model, while the remaining $K - 1$ subsets are used to train the model. In this evaluation, K

is set to 10 as suggested by Kohavi (Kohavi, 1995) to reliably demonstrate the appropriateness of a proposed predictive model. All experiments are conducted in Matlab v12 on a PC with 2.53 GHz CPU and 4 GB memory.

5.1. Accuracy metrics

Several metrics have been used to evaluate anomaly detection approaches. In this evaluation, the *precision*, *recall* and *F-score* well-known metrics are used to quantitatively measure the performance of a system, as they are not dependent on the size of the training and testing data set. The metrics used are defined as follows:

$$Recall\ (Detection\ rate) = \frac{TP}{TP + FN} \tag{11}$$

$$False\ positive\ rate = \frac{FP}{FP + TN} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$F - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{14}$$

where *TP* is the number of inconsistent observations that are correctly detected, *FN* is the number of inconsistent observations that have occurred but have not been detected, *FP* is the number of consistent observations that have been incorrectly flagged as inconsistent, and *TN* is the number of consistent observations that have been correctly classified. The recall (detection rate) represents the proportion of correctly detected inconsistent observations to the actual size of the inconsistent observations in the testing data set, while false positive rate is the proportion of the consistent observations incorrectly classified as inconsistent to the actual size of the consistent observations in the testing data set. The precision metric is used to demonstrate the robustness of the IDS in minimizing the false positive rate. However, the system can obtain a high precision score even though a number of inconsistent observations have been missed. Similarly, the system can obtain a high recall score although the false positive rate is higher. Therefore, the F-score, which is the harmonic mean of precision and recall, would be a more appropriate metric to demonstrate the accuracy of anomaly detection approaches.

5.2. Performance evaluation

Two parts are discussed this section: the first part is the accuracy of the identification of consistent and inconsistent observations as a first phase to extract detection rules, while the second part is the detection accuracy of these extracted detection rules, and the impact of the number of learned rules on detection accuracy and the time elapsed for calculating the cosine similarity of a test observation against these rules.

5.2.1. Identification accuracy

As previously discussed, the proposed approach initially identifies the inconsistent observations from the training data set to form two data sets, inconsistent and consistent, from which the detection rules are generated. Due to the inconsistency score for each observation based on its neighbourhood density in this phase, the value of k-nearest neighbours needs to be given. In this evaluation, the value of k-nearest is set to be 5% of the representative data set, as this value is assumed as the maximum percentage of anomalies in a data set (Eskin et al., 2002). Therefore, this value is assumed to discriminate between inconsistent observations and consistent ones in terms of neighbourhood density. Table 1 shows promising results when identifying the inconsistent observations from the consistent ones on all data sets.

5.2.2. Detection accuracy and efficiency

Firstly, the proposed approach gives an inconsistency score for each observation in the training data set using Algorithm 1, and then separates the inconsistent observations from consistent ones. Secondly, it adopts the fixed-width clustering algorithm to individually cluster the inconsistent and consistent observations into micro-clusters with constant-width w as discussed earlier in Section 3.2. The clusters created from consistent observations are labelled as normal, while the clusters created from the inconsistent observations are labelled as abnormal. The cluster-width parameter w is statistically determined and calculated as in Equation (6). In the proposed approach, one parameter is required, which is the k-nearest-neighbours. However, this parameter can be heuristically determined as discussed earlier in Section 4. The value of parameter w (as discussed in Section 3.3) controls the number of learned rules. A small value of the parameter w will result in many rules and, consequently, a high similarity computation is required in the detection phase, while, a large will result in few rules that may not mostly represent the learning data set, and therefore the detection accuracy of the proposed approach may be degraded. Tables 2–4 demonstrate the detection accuracy of learned rules, and the impact of their sizes on detection accuracy and the amount of time required to calculate the cosine similarity of a test observation.

As can be seen in Tables 2–4, the various values of the cluster width parameter w are used to learn the proximity-detection rules, where these rules are assumed to represent the behaviours (either consistent or inconsistent) of the training data set. Each table demonstrates the detection accuracy results for a particular data set, where each row of the table represents detection accuracy results of the learned proximity-detection rules for a specific value of the parameter w, and also shows how this parameter controls the size of rules of which detection accuracy and the amount of time required to judge any observations against these rules are affected positively or negatively. For instance, the first row in

Table 1 – The accuracy results in identifying consistent/inconsistent observations on the three data sets.

Data set	Detection rate	False positive	Precision	F-score
DUWWTP	92.86%	0.19%	92.86%	92.86%
SimData1	98.04%	0.17%	84.75%	90.91%
SimData2	100.00%	0.15%	86.21%	92.59%

Table 2 – The accuracy results of the proposed approach on the real SCADA data set (DUWWTP).

w	#Obs	#Rules	Avg.Time (ms)	Detection rate	False positive	Precision	F-score
$\mu = 0.7491$	476	67	0.1237	85.71%	0.00%	100.00%	92.31%
$\mu + 0.5\sigma = 0.8551$	476	42	0.1355	78.57%	0.00%	100.00%	88.00%
$\mu + \sigma = 0.9612$	476	26	0.0555	71.43%	0.00%	100.00%	83.33%
$\mu + 1.5\sigma = 1.0672$	476	18	0.0434	57.14%	0.00%	100.00%	72.73%
$\mu + 2\sigma = 1.1732$	476	17	0.0415	50.00%	5.88%	70.00%	58.33%
$\mu + 2.5\sigma = 1.2793$	476	14	0.0434	50.00%	7.84%	63.64%	56.00%
$\mu + 3\sigma = 1.3853$	476	10	0.0316	42.86%	13.73%	46.15%	44.44%
$\mu + 3.5\sigma = 1.4913$	476	9	0.0313	42.86%	15.69%	42.86%	42.86%
$\mu + 4\sigma = 1.5973$	476	7	0.0251	42.86%	17.65%	40.00%	41.38%
$\mu + 4.5\sigma = 1.7034$	476	5	0.0232	35.71%	23.53%	29.41%	32.26%
$\mu + 5\sigma = 1.8094$	476	5	0.0223	35.71%	29.41%	25.00%	29.41%

w: The cluster width parameter, μ and σ are the mean and standard deviation of inconsistency scores for all observations in the training data set respectively.
Obs: The number of observations in the training Data set.
Avg.Time: the time spent for calculating the cosine similarity of a test observation against learned rules.

Table 2 indicates that the value of the parameter w is set to 0.7506, that is the mean of inconsistency scores of all training observations, to create micro-clusters for both consistent/inconsistent observations. In this example, 67 micro-clusters are created whose centroids are then used for the proximity-detection rules.

Overall, the accuracy of proximity-detection rules on the three data sets (the real and simulated ones) demonstrate promising results when the value of parameter w is set to the mean of inconsistency scores. It is important to note that the learned proximity-detection rules mostly represent both consistent/inconsistent behaviours in the training data sets. In addition, the small size of these rules significantly reduces the time required in the detection phase to judge whether an observation is consistent or inconsistent. It can be seen in Table 3 that from 9461 observations, 348 proximity-detection rules are learned, and less than 1 millisecond is required to predict the behaviour of the coming observation.

5.3. *Performance evaluation of unsupervised approaches*

As the proposed approach is based on the unsupervised mode, we chose *K*-means (MacQueen, 1967) and fixed-width (Eskin et al., 2002), clustering-based algorithms (Jianliang et al., 2009; Münz et al., 2007; Oldmeadow et al., 2004; Eskin et al., 2002), as they are promising techniques for learning unsupervised clustering-based anomaly detection models.

5.3.1. *Fixed-width clustering algorithm*

This algorithm creates a set of clusters of fixed width and the various clustering steps of this algorithm are discussed in Section 3.3. To evaluate the detection accuracy of this algorithm, we used the approach in (Eskin et al., 2002) that performs detection by clustering the training data set into a number of clusters with constant width, and label each cluster as either *normal* or *abnormal* by taking a percentage *N* of the largest clusters to be labelled as normal, while the rest are considered as abnormal. In a slightly different way, we labelled any cluster as abnormal when the proportion of the observations associated with it to the actual number of the observations in training data set, is less than the predefined-percentage *N*; otherwise, the cluster is labelled as normal. During the testing phase, an observation is judged by the label of the closest cluster. Clearly, two main user-specified parameters are required: cluster-width and *N*. However, it is challenging to directly determine the appropriate parameters that produce a detection model that maximizes the detection accuracy. Therefore, we attempted ten values of cluster-width from 0.01 to 0.2 increased by 0.02, and each cluster-width is tested with ten values of *N* from 1 to 10. For simplicity, we chose the top ten F-scores, as shown in Tables 5–7.

Table 3 – The accuracy results of the proposed approach on the simulated SCADA data set (SimData1).

w	#Obs	#Rules	Avg.Time (ms)	Detection rate	False positive	Precision	F-score
$M = 0.1455$	9461	348	0.5033	98.04%	0.48%	95.24%	96.62%
$M + 0.5\sigma = 0.2141$	9461	211	0.3110	98.04%	0.77%	92.59%	95.24%
$M + \sigma = 0.2827$	9461	145	0.2262	93.14%	1.25%	87.96%	90.48%
$M + 1.5\sigma = 0.3513$	9461	108	0.1621	90.20%	1.25%	87.62%	88.89%
$M + 2\sigma = 0.4199$	9461	81	0.1271	78.43%	1.63%	82.47%	80.40%
$M + 2.5\sigma = 0.4885$	9461	60	0.1067	73.53%	1.92%	78.95%	76.14%
$M + 3\sigma = 0.5571$	9461	37	0.0657	63.73%	2.40%	72.22%	67.71%
$M + 3.5\sigma = 0.6257$	9461	30	0.0563	60.78%	2.60%	69.66%	64.92%
$M + 4\sigma = 0.6943$	9461	19	0.0420	57.84%	2.88%	66.29%	61.78%
$M + 4.5\sigma = 0.7629$	9461	16	0.0365	49.02%	3.56%	57.47%	52.91%
$M + 5\sigma = 0.8315$	9461	13	0.0331	47.06%	3.85%	54.55%	50.53%

Table 4 – The accuracy results of the proposed approach on the simulated SCADA data set (SimData2).

w	#Obs	#Rules	Avg.Time (ms)	Detection rate	False positive	Precision	F-score
$M = 0.1473$	9461	351	0.5543	98.00%	0.19%	98.00%	98.00%
$M + 0.5\sigma = 0.2183$	9461	204	0.3102	95.00%	0.48%	95.00%	95.00%
$M + \sigma = 0.2893$	9461	142	0.2140	85.00%	0.77%	91.40%	88.08%
$\mu + 1.5\sigma = 0.3603$	9461	103	0.1686	77.00%	0.96%	88.51%	82.35%
$\mu + 2\sigma = 0.4313$	9461	83	0.1305	74.00%	1.25%	85.06%	79.14%
$\mu + 2.5\sigma = 0.5022$	9461	58	0.1097	69.00%	1.54%	81.18%	74.59%
$\mu + 3\sigma = 0.5732$	9461	37	0.0697	65.00%	1.92%	76.47%	70.27%
$\mu + 3.5\sigma = 0.6442$	9461	24	0.0480	40.00%	2.31%	62.50%	48.78%
$\mu + 4\sigma = 0.7152$	9461	21	0.0473	30.00%	2.40%	54.55%	38.71%
$\mu + 4.5\sigma = 0.7862$	9461	15	0.0379	25.00%	2.69%	47.17%	32.68%
$\mu + 5\sigma = 0.8571$	9461	13	0.0318	20.00%	2.88%	40.00%	26.67%

As it can be seen in Tables 5–7, the fixed-width clustering algorithm performs well in detecting inconsistent observations provided that the two main user-specified parameters, as discussed, are appropriately chosen. However, determining the best parameters with unlabelled data is impractical. In addition, each data set has different parameters to maximize the detection accuracy.

5.3.2. *k-means algorithm*

k-means (MacQueen, 1967) is a clustering algorithm which is simple and commonly used to partition data into a number of clusters, where the number of clusters must be specified in advance. This algorithm involves several steps to cluster the data:

1. Define the number of clusters *C*.
2. Randomly select *C* of the objects in the data set, and consider them as the clusters' centroids.
3. Assign each object in the data set to the nearest cluster.
4. Recalculate the new centroid for each cluster by computing the mean for all objects in each cluster.
5. Repeat step 3 until the change of centroids is stable.

Similarly, with *k*-means, we used the same method as was used with the fixed-width clustering algorithm to build the detection model, although the number of clusters is specified instead of cluster-width. We used ten different values of the number of clusters *C*, from 1 to 10, and each different value is tested with ten values of *N* from 1 to 10. Again, for simplicity, we demonstrated the top 10 *F*-scores, as shown in Tables 11–13.

The top ten accuracy results of *k*-means in Tables 11–13 demonstrate that the *k*-means algorithm performs worse than the fixed-width clustering algorithm. Similar to the fixed-width algorithm, *k*-means requires two main user-specified parameters: the number of clusters *C* and the percentage of the number of observations in each cluster to be labelled as abnormal *N*. Moreover, it can be seen that all data sets have different parameters so as to maximize the detection accuracy.

Table 5 – The accuracy results of fixed-width clustering based approach on the real SCADA data set (DUWWTP).

w	N	Detection rate	False positive	Precision	F-score
0.03	1	98.10%	8.51%	62.85%	76.15%
0.05	1	75.71%	2.79%	80.53%	76.07%
0.05	2	77.62%	5.00%	72.35%	72.24%
0.07	1	59.52%	1.10%	89.92%	69.56%
0.07	4	59.05%	1.11%	89.45%	69.18%
0.09	2	60.00%	1.50%	86.21%	68.54%
0.09	6	60.00%	1.56%	85.02%	68.31%
0.09	10	60.00%	1.56%	85.76%	68.26%
0.09	8	60.00%	1.62%	83.39%	68.17%
0.09	5	60.00%	1.63%	84.91%	68.12%

w: The cluster width parameter.
N: The percentage of the data in a cluster to be assumed as malicious.

5.4. *Performance of the semi-supervised approach*

A supervised anomaly detection learning approach requires class labels for both normal and abnormal behaviours to learn anomaly detection models, but, by contrast, a semi-supervised learning approach can learn such models using either normal or abnormal behaviour. For instance, if the training data set is assumed to represent the normal behaviour, the artificial data can be generated to take the role of the abnormal behaviour. In this evaluation, following the work of Hempstalk et al. (Hempstalk et al., 2008), artificial data are generated by using a density estimator to form a reference distribution from the training data, that represents one behaviour. Then, the reference distribution is used to generate artificial data that represents the other behaviour. Afterwards, we a Naive Bayes (John and Langley, 1995) is used to learn the

Table 6 – The accuracy results of fixed-width clustering based approach on the real SCADA data set (DUWWTP).

w	N	Detection rate	False positive	Precision	F-score
0.11	1	96.47%	0.01%	99.80%	98.11%
0.13	1	96.47%	0.01%	99.80%	98.11%
0.13	3	96.47%	0.01%	99.80%	98.11%
0.15	1	96.47%	0.01%	99.80%	98.11%
0.15	2	96.47%	0.01%	99.80%	98.11%
0.15	3	96.47%	0.01%	99.80%	98.11%
0.11	2	96.47%	0.01%	99.80%	98.10%
0.11	3	96.47%	0.01%	99.80%	98.10%
0.13	2	96.47%	0.01%	99.80%	98.10%
0.09	2	96.47%	0.10%	97.98%	97.21%

Table 7 – The accuracy results of fixed-width clustering based approach on the real SCADA data set (DUWWTP).

w	N	Detection rate	False positive	Precision	F-score
0.11	1	100.00%	0.01%	99.80%	99.90%
0.11	2	100.00%	0.01%	99.80%	99.90%
0.11	3	100.00%	0.01%	99.80%	99.90%
0.11	4	100.00%	0.01%	99.80%	99.90%
0.13	2	99.67%	0.01%	99.80%	99.73%
0.09	4	100.00%	0.12%	97.70%	98.83%
0.09	2	100.00%	0.12%	97.69%	98.82%
0.09	1	100.00%	0.12%	97.51%	98.73%
0.09	3	100.00%	0.13%	97.47%	98.70%
0.07	3	100.00%	0.14%	97.15%	98.54%

anomaly detection model, as it provides the most accurate results among a number of classifiers that have been investigated in this evaluation.

For this evaluation WEKA data mining software (Developer version 3.7.10 (Hall et al., 2009)) is used. This is because the artificial data generation approach (Hempstalk et al., 2008) and a number of supervised classifiers such as, Naive Bayes (John and Langley, 1995), are implemented in this software. The population of inconsistent observations is set to 50%. The consistent rejection rate, where a new observation is considered as consistent, is experimentally determined and the optimal value is found to be different for each data set.

A semi-supervised learning approach appears to be a time- and cost-efficient technique, since domain experts are not required to label a hundred thousand of data observations. With this technique, only the targeted system is required to operate for a long time under normal condition in order to obtain purely normal data that comprehensively represent normal behaviours. However, it cannot be guaranteed that no anomalous activity will occur during the data collection period. To demonstrate this issue, five anomaly detection models are learned for each data set, as shown in Tables 8–10. The first model is learned from a training data set containing only consistent observations, while the others are learned from the same training data set, but each time a number of inconsistent observations are introduced into the training data set as consistent in order to observe their impact on the detection accuracy of a learned model. It is important to note, as shown in Tables 8–10 that the detection accuracy degraded in semi-supervised learning when some inconsistent observations are learned as consistent during the training phase. Clearly, the rate of degradation increased, as the amount of inconsistent observations that are added into the training data as consistent is increased.

Table 8 – The accuracy results of semi-supervised approach on the real SCADA data set (DUWWTP) with consistent rejection rate = 0.002.

N	InC	Detection rate	False positive	Precision	F-score
0%	0	92.86%	1.18%	95.59%	94.20%
10%	1	90.71%	1.76%	93.38%	92.03%
20%	3	77.62%	0.78%	96.45%	86.02%
30%	4	71.79%	0.59%	97.10%	82.55%
40%	6	64.57%	0.47%	97.41%	77.66%

N: The percentage of inconsistent observations that are labelled as consistent at the training time.
InC: The number of inconsistent observations that are added into the training data set as consistent ones.

Table 9 – The accuracy results of semi-supervised approach on the simulated SCADA data set (SimData1) with consistent rejection rate = 0.05.

N	InC	Detection rate	False positive	Precision	F-score
0%	0	98.04%	5.39%	78.13%	86.96%
10%	10	40.22%	5.67%	55.22%	46.54%
20%	20	36.59%	8.16%	40.54%	38.46%
30%	31	23.94%	9.27%	25.00%	24.46%
40%	41	8.20%	9.29%	8.77%	8.47%

Table 10 – The accuracy results of semi-supervised approach on the simulated SCADA data set (SimData2) with consistent rejection rate = 0.15.

N	InC	Detection rate	False positive	Precision	F-score
0%	0	75.00%	13.46%	34.88%	47.62%
10%	10	67.78%	13.90%	29.47%	41.08%
20%	20	58.75%	14.43%	23.50%	33.57%
30%	30	54.29%	13.83%	20.43%	29.69%
40%	40	46.67%	13.52%	16.09%	23.93%

6. Conclusion

In this paper, we proposed an innovative unsupervised SCADA data-driven anomaly detection approach to detect integrity attacks tailored to SCADA systems. This has been done by initially identifying the consistent and inconsistent states of SCADA data automatically, and then also automatically extracting proximity-based detection rules from the identified states to detect inconsistent states. Experimental results show the ability of the proposed approach to automatically identify consistent and inconsistent states, with significant accuracy. Moreover, the automatically extracted proximity-based detection rules show promising detection

Table 11 – The accuracy results of k-means clustering based approach on the real SCADA data set (DUWWTP).

C	N	Detection rate	False positive	Precision	F-score
10	7	71.43%	6.05%	62.75%	65.40%
8	10	70.95%	6.83%	65.46%	64.42%
10	8	72.86%	9.51%	58.37%	61.19%
15	4	66.67%	7.29%	58.43%	60.93%
13	4	58.10%	3.77%	71.02%	60.56%
13	5	67.62%	6.69%	59.41%	60.56%
9	9	74.76%	10.77%	53.19%	60.48%
12	7	72.86%	9.96%	54.10%	60.29%
15	5	74.76%	10.73%	51.52%	59.45%
14	5	67.62%	8.06%	55.26%	59.40%

C: The number of created clusters.
N: The percentage of the data in a cluster to be assumed as malicious.

Table 12 – The accuracy results of k-means clustering based approach on the real SCADA data set (DUWWTP).

w	N	Detection rate	False positive	Precision	F-score
4	5	99.61%	4.12%	54.60%	70.43%
3	10	99.61%	4.12%	54.50%	70.39%
3	8	99.61%	4.12%	54.50%	70.38%
4	6	99.61%	4.12%	54.48%	70.38%
6	6	99.61%	4.12%	54.49%	70.38%
5	6	99.61%	4.12%	54.47%	70.37%
6	5	99.54%	4.55%	53.45%	69.27%
5	5	99.61%	4.58%	53.40%	69.23%
7	6	99.61%	4.70%	52.97%	68.76%
4	10	99.61%	5.05%	52.43%	68.12%

Table 13 – The accuracy results of k-means clustering based approach on the real SCADA data set (DUWWTP).

C	N	Detection rate	False positive	Precision	F-score
15	4	32.07%	7.02%	20.06%	23.50%
15	2	23.67%	0.63%	23.00%	23.23%
15	3	28.13%	2.22%	19.96%	22.59%
14	3	21.00%	1.11%	17.80%	18.84%
13	1	16.67%	0.02%	16.60%	16.63%
15	1	16.67%	0.04%	16.54%	16.60%
14	1	16.67%	0.04%	15.98%	16.28%
15	6	57.47%	33.33%	7.92%	13.87%
9	2	13.33%	0.12%	13.33%	13.33%
12	1	13.33%	0.01%	13.33%	13.33%

accuracy results compared to three existing anomaly detection approaches, two of which are based on unsupervised learning, while the third is based on semi-supervised learning. However, the proposed approach is based on the k-nearest neighbour technique, which is computationally expensive when computing an inconsistency score for each observation. Thus, the reduction of computational time will be studied in the future. Moreover, we will evaluate this approach with further anomaly detection approaches in various application domains.

REFERENCES

Alcaraz C, Lopez J. WASAM: a dynamic wide-area situational awareness model for critical domains in smart grids. Future Gener Comput Syst 2014;30:146–54.

Alcaraz C, Lopez J. Diagnosis mechanism for accurate monitoring in critical infrastructure protection. Comput Stand Interfaces 2014;36(3):501–12.

Almalawi A, Tari Z, K. I, Fahad Adil. SCADAVT – a framework for SCADA security testbed based on virtualization technology. In: IEEE 38th Conference on Local Computer Networks (LCN); 2013. p. 639–46.

Angiulli F, Pizzuti C. Outlier mining in large high-dimensional data sets. IEEE Trans Knowl Data Eng 2005;17(2):203–15.

Ankerst M, Breunig MM, Kriegel H-P, Sander J. Optics:ordering points to identify the clustering structure. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, vol. 28; 1999. p. 49–60.

Arning A, Agrawal R, Raghavan P. A linear method for deviation detection in large databases. In: Proceedings of 2nd Conference on Knowledge Discovery and Data Mining (KDD); 1996. p. 164–9.

Beygelzimer A, Kakade S, Langford J. Cover trees for nearest neighbor. In: Proceedings of the 23rd International Conference on Machine learning; 2006. p. 97–104.

Breunig MM, Kriegel H-P, Ng RT, Sander J. Optics-of: identifying local outliers. In: Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery; 1999. p. 262–70.

Breunig MM, Kriegel H-P, Ng RT, Sander J. Lof: identifying density-based local outliers. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, vol. 29; 2000. p. 93–104.

Carcano A, Coletta A, Guglielmi M, Masera M, Fovino IN, Trombetta A. A multidimensional critical state analysis for detecting intrusions in SCADA systems. IEEE Trans Ind Inform 2011;7(2):179–86.

Cheung S, Dutertre B, Fong M, Lindqvist U, Skinner K, Valdes A. Using model-based intrusion detection for SCADA networks. In: Proceedings of the SCADA Security Scientific Symposium; 2007. p. 1–12.

Digitalbond. IDS-signatures of Modbus/TCP; 2013. http://www.digitalbond.com/index.php/research/ids-signatures/modbus-tcp-ids-signatures [Date last accessed 15.08.13] (2013).

Eskin E, Arnold A, Prerau M, Portnoy L, Stolfo S. A geometric framework for unsupervised anomaly detection. In: Applications of data mining in computer security. Springer; 2002. p. 77–101.

Fahad A, Tari Z, Khalil I, Habib I, Alnuweiri H. Toward an efficient and scalable feature selection approach for internet traffic classification. Comput Netw 2013;2:237–52.

Fahad A, Tari Z, Almalawi A, Goscinski A, Khalil I, Mahmood A. Ppfscada: privacy preserving framework for scada data publishing. Future Gener Comput Syst 2014;37:496–511.

Falliere N, Murchu LO, Chien E. W32. stuxnet dossier: version 1.4. Tech. rep. CA: Symantec; 2011. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf [Date last accessed 10.7.13]

Fernandez EB, Larrondo-Petrie MM. Designing secure scada systems using security patterns. In: 43rd Hawaii International Conference on System Sciences(HICSS); 2010. p. 1–8.

Fovino IN, Carcano A, Murel TDL, Trombetta A, Masera M. Modbus/DNP3 state-based intrusion detection system. In: Proceedings of the 24th International Conference on Advanced Information Networking and Applications (AINA); 2010. p. 729–36.

Fovino IN, Coletta A, Carcano A, Masera M. Critical state-based filtering system for securing SCADA network protocols. IEEE Trans Ind Electron 2012;59(10):3943–50.

Frank A, Asuncion A. UCI machine learning repository; 2013. http://archive.ics.uci.edu/ml [Date last accessed 10.07.13].

Fukunaga K, Narendra PM. A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans Comput 1975;100(7):750–3.

Gao W, Morris T, Reaves B, Richey D. On scada control system command and response injection and intrusion detection. In: eCrime Researchers Summit (eCrime); 2010. p. 1–9.

Gross P, Parekh J, Kaiser G. Secure selecticast for collaborative intrusion detection systems. In: Proceedings of the 3rd International Workshop on Distributed Event-Based Systems (DEBS); 2004. p. 50–5.

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The weka data mining software: an update. ACM SIGKDD Explor Newsl 2009;11(1):10–8.

Hempstalk K, Frank E, Witten IH. One-class classification by combining density and class probability estimation. In: Machine learning and knowledge discovery in databases; 2008. p. 505–19.
M. IDA. Modbus messaging on TCP/IP implementation guide v1.0a; 2013. http://www.modbus.org/specs.php [Date last accessed 15.03.13].
Jianliang M, Haikun S, Ling B. The application on intrusion detection based on k-means cluster algorithm. In: 9 International Forum on Information Technology and Applications (IFITA), vol. 1; 2009. p. 150–2.
Jin X, Bigham J, Rodaway J, Gamez D, Phillips C. Anomaly detection in electricity cyber infrastructures. In: Proceedings of the International Workshop on Complex Networks and Infrastructure Protection (CNIP); 2006. p. 1–12.
John GH, Langley P. Estimating continuous distributions in bayesian classifiers. In: Proceedings of the eleventh conference on uncertainty in artificial intelligence; 1995. p. 338–45.
Johnson RA, Wichern DW. Applied multivariate statistical analysis. NJ: Prentice hall Upper Saddle River; 2002.
Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) vol. 14; 1995. p. 1137–45.
Kumar SAP, Kumar A, Srinivasan S. Statistical based intrusion detection framework using six sigma technique. Int J Comput Sci Netw Secur (IJCSNS) 2007;7(10):333–42.
Lewis A. The epanet programmers toolkit for analysis of water distribution systems; 2011. http://www.epa.gov/nrmrl/wswrd/dw/epanet.html [Date last accessed 15.08.13].
Linda O, Vollmer T, Manic M. Neural network based intrusion detection system for critical infrastructures. In: Proceedings of International Joint Conference on Neural Networks; 2009. p. 1827–34.
MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1; 1967. p. 281–97.
Marton I, Sánchezb AI, Carlosa S, Martorella S. Application of data driven methods for condition monitoring maintenance. Chem Eng Trans 2013;33:301–6.
Melbourne Water. Daily residential water use for Melbourne; 2009. http://www.melbournewater.com.au [Date last accessed 15.08.13].
Münz G, Li S, Carle G. Traffic anomaly detection using k-means clustering. In: Proceedings of 5th GI/ITG KuVS Workshop on Future Internet, vol. 4; 2007. p. 1–8.
Nicholson A, Webber S, Dyer S, Patel T, Janicke H. Scada security in the light of cyber-warfare. Comput Secur 2012;31(4):418–36.
Ning P, Cui Y, Reeves DS. Constructing attack scenarios through correlation of intrusion alerts. In: Proceedings of the 9th ACM conference on Computer and communications security; 2002. p. 245–54.
Oldmeadow J, Ravinutala S, Leckie C. Adaptive clustering for network intrusion detection, advances in knowledge discovery and data Mining; 2004. p. 255–9.
Oman P, Schweitzer E, Frincke D. Concerns about intrusions into remotely accessible substation controllers and scada systems. In: Proceedings of the Twenty-Seventh Annual Western Protective Relay Conference, vol. 160; 2000.
Queiroz C, Mahmood A, Tari Z. SCADASima framework for building SCADA simulations. IEEE Trans Smart Grid 2011;2(4):589–97.
Rrushi J. Composite intrusion detection in process control networks [Ph.D. thesis]. University of Milano; April 2009.
Slay J, Miller M. Lessons learned from the maroochy water breach. Crit Infrastruct Prot 2007:73–82.
Sproull RF. Refinements to nearest-neighbor searching in k-dimensional trees. Algorithmica 1991;6(1):579–89.
Steinbach M, Karypis G, Kumar V. A comparison of document clustering techniques. In: Proceedings of the KDD workshop on text mining; 2000. p. 525–6.
Team. Modbus implementation; 2012. http://code.google.com/p/pymodbus [Date last accessed 15.08.13].
Valdes A, Cheung S. Communication pattern anomaly detection in process control systems. In: Proceedings of Conference on Technologies for Homeland Security (HST); 2009. p. 22–9.
Verba J, Milvich M. Idaho national laboratory supervisory control and data acquisition intrusion detection system (SCADA IDS). In: Proceedings of IEEE Conference on Technologies for Homeland Security (HST); 2008. p. 469–73.
Vydunas Šaltenis. Outlier detection based on the distribution of distances between data points. Informatica 2004;15(3):399–410.
Wei D, Lu Y, Jafari M, Skare PM, Rohde K. Protecting smart grid automation systems against cyberattacks. IEEE Trans Smart Grid 2011;2(4):782–95.
Wenxian Y, Jiesheng J. Wind turbine condition monitoring and reliability analysis by scada information. In: 2nd International Conference on Mechanic Automation and Control Engineering (MACE); 2011. p. 1872–5.
Yang D, Usynin A, Hines JW. Anomaly-based intrusion detection for scada systems. In: 5th Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (NPIC&HMIT 05), Citeseer; 2006. p. 12–6.
Yu X, Cecati C, Dillon T, Simoes MG. The new frontier of smart grids. IEEE Ind Electron Mag 2011;5(3):49–63.
Zaher A, McArthur S, Infield D, Patel Y. Online wind turbine fault detection through automated scada data analysis. Wind Energy 2009;12(6):574–93.

Abdulmohsen Almalawi received his B.S. degree in Computer Science from King Abdul Aziz University, Jeddah, Saudi Arabia, in 2003. He received his M.S. degree in 2008 from RMIT University, Melbourne, Australia, and he is currently a Ph.D. candidate in the Department of Computer Science and Information Technology at the University of RMIT. His research interests are in the areas of machine learning, and SCADA security.

Xinghuo Yu Xinghuo Yu is currently with the RMIT University, Melbourne, Australia, where he is the Director of the RMIT Platform Technologies Research Institute. He has published over 350 refereed papers in technical journals, books, and conference proceedings. His research interests include variable structure and nonlinear control, complex and intelligent systems, and industrial applications. Prof. Yu is a Fellow of the Institution of Engineers Australia and the Australian Computer Society. He is currently serving as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART I, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.

Zahir Tari is a full professor at RMIT University. He is also the direction of the DSN (Distributed Systems and Networking) discipline at the School of Computer Science and IT, RMIT (Australia). His main research areas are in performance and security in various areas of application (e.g. Web servers, Content Delivery Networks, SCADA systems etc). Prof Tari regularly publishes in reputable journals and conferences. He acted as the program committee chair as well as general chair over fifteen international conferences (e.g. DOA, CoopIS, ODBASE, GADA, IFIP DS 11.3 on Database Security). He is also the co-author of a few books. He has also been General Chair of more than 12 conferences. He is the recipient of 14 Australian Research Council (ARC) grants. More details about Zahir and his team can be found at http://www.cs.rmit.edu.au/dsn.

Adil Fahad received his B.S. degree in Computer Science from King Abdul Aziz University, Jeddah, Saudi Arabia, in 2003. He received his M.S. degree (with high distinction) in 2008 from RMIT University, Melbourne, Australia, and he is currently a Ph.D. candidate in the Department of Computer Science and Information Technology at the University of RMIT. He joined the University of Albaha as a lecturer in 2009 and took a leave of absence in 2010 for his Ph.D. studies. His research interests are in the areas of wireless sensor networks, mobile networks, SCADA security and ad-hoc networks with emphasis on data mining, statistical analysis/modelling and machine learning.

Ibrahim Khalil received the Ph.D. degree from the University of Berne, Berne, Switzerland, in 2003. He is a Senior Lecturer in the School of Computer Science and IT, RMIT University, Melbourne, Australia. He has several years of experience in Silicon Valley-based companies working on Large Network Provisioning and Management software. He also worked as an academic in several research universities. Before joining RMIT, he worked for EPFL and University of Berne in Switzerland and Osaka University in Japan. His research interests are quality of service, wireless sensor networks, and remote healthcare.