Multi-level Anomaly Detection in Industrial Control Systems via Package Signatures and LSTM networks

Cheng Feng, Tingting Li and Deeph Chana
Institute for Security Science and Technology
Imperial College London
London, United Kingdom
Email: {c.feng, tingting.li, d.chana}@imperial.ac.uk

***Abstract*—We outline an anomaly detection method for industrial control systems (ICS) that combines the analysis of network package contents that are transacted between ICS nodes and their time-series structure. Specifically, we take advantage of the predictable and regular nature of communication patterns that exist between so-called *field devices* in ICS networks. By observing a system for a period of time without the presence of anomalies we develop a base-line signature database for general packages. A Bloom filter is used to store the signature database which is then used for package content level anomaly detection. Furthermore, we approach time-series anomaly detection by proposing a stacked Long Short Term Memory (LSTM) network-based softmax classifier which learns to predict the most likely package signatures that are likely to occur given previously seen package traffic. Finally, by the inspection of a real dataset created from a gas pipeline SCADA system, we show that an anomaly detection scheme combining both approaches can achieve higher performance compared to various current state-of-the-art techniques.**

***Keywords*—industrial control systems; anomaly detection; signature database; long short term memory networks; Bloom filters;**

I. Introduction

Composed of combinations of hardware, software, networks and operators, industrial control systems (ICS) orchestrate and control the myriad of functions needed to execute complex tasks such as the delivery of utility services and the execution of intricate and disparate manufacturing processes. The range of ICS application scenarios include water treatment[1], gas pipelines [2], power plants [3] and industrial manufacture [4]. Depending on implementation specifics, ICS instances are typically described as combinations of supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and programmable logic controllers (PLC).

Traditional ICS are not networked and hence are considered to be well protected by so-called "air-gapped" separation. To further promote efficient remote-control and higher throughput, smart ICT technologies have been widely merged into ICS where most of components are old, originally insecure-by-design and hard to upgrade. Such evolution of ICS builds up a connection between physical and cyber worlds, but also makes them vulnerable to cyber attacks. NIST summarises the main security issues posed to modern ICS in [5], including insecure-by-design communication protocols [6], insecure network segregation and access controls [7], the lack of ICS-specific firewalls and anomaly detection systems [8].

Cyber attacks against ICS would lead to disruption to controlling those critical infrastructures and result in harmful physical damage to plants, environment and humans [9]. According to ICS-CERT, the ICS-targeted attacks have been continuously increasing in the past few years. There were 73 incidents reported to ICS-CERT by trusted industrial partners in 2013 [10], then 245 reported in 2014 [11] and 295 incidents in 2015 [12].

The typical architecture of modern ICS roughly consists of three parts : (i) a *Corporate Network* provides business-to-business and business-to-customer services, which is often directly exposed to the Internet, with VPN access, email and web servers. (ii) a *Control Network* receives and processes data from field devices, and responses with proper control commands. Human Machine Interfaces (HMIs), control workstations, alarm and anomaly detection systems are often situated in this zone. (iii) a *Field Network* that is the fusion of PLCs, actuators and sensors for measurement data transmission and the direct control of industrial processes. Cyber attacks against ICS often start with gaining access to the *Corporate Network* via social engineering based methods, then propagate virus across the whole network in search of valuable targets (e.g. assets in *Control Network* with direct access to field devices), and finally sabotage control programs running on field devices. The first widely documented ICS cyber attack was *Stuxnet*, disclosed in 2011 [13]. Here the virus was introduced by a removal drive, and eventually managed to change the program controlling field devices. Two more recent examples are the security breach to a German steel mill initiated by spear-phishing emails in 2014 [14], and the attack leading to massive outage for Ukrainian power companies in 2015 [15].

Intrusion (or anomaly) detection systems (IDS) are able to monitor the network activities from data logs or network traffic, and effectively generate alarms of potential or attempted intrusions. Although this topic has been extensively studied in conventional IT security community, limited effort has been conducted to develop ICS-specific anomaly detection systems. Accurate anomaly detection systems are able to further facilitate fast accident-response mechanism, and also initiate the interaction between security and control practitioners such that potential security breaches can be found more quickly and accurately. Therefore, there is an urgent need of effective ICS-specific anomaly detection systems.

Anomaly detection in ICS is a challenging problem for the following reasons: (i) ICS-specific communication protocols (e.g. DNP3, Modbus) are not considered by conventional

IDS. (ii) the lack of real-world ICS datasets for training and evaluation of anomaly detectors. (iii) anomaly detection for ICS cannot solely depend on network protocol information; additional information related to physical process control also need to be examined. This significantly increases the dimensionality and complexity of data samples. (iv) physical process control variables may exhibit noisy behaviours by nature, which is likely to result in high false positive rates for anomaly detectors and low detection rates of attacks.

ICS-specific IDS has been an active topic for several years, and there exists some work in the area. Conventional intrusion detection methods have been applied such as the model-based approach [16] and the behaviours-based approach [17]. These works specifically incorporated ICS protocols in IDS (e.g. Modbus/DNP3 [18], IEC standards [19] [20]). A detailed discussion about them can be found in the next related work section. However, there are several limitations with most existing work: (i) Most methods rely highly on predefined models and signatures to detect anomalous behaviours, requiring massive human effort at the preliminary stage. (ii) Such models and signatures are often constructed from known attacks and hence are not capable of detecting unknown or zero-day attacks. (iii) The existing IDS methods are mostly tailored for specific protocols and systems, which lack sufficient generality and flexibility to adapt to other systems. (iv) The temporal dependencies between packages have been least studied in the literature, which, however, is of great help to identify advanced persistent attacks and collective anomalies.

To address the above issues, we have developed a generalized multi-level anomaly detection framework based on network package signatures and machine learning techniques to enable the construction of an ICS-specific IDS with highly reduced human input. Moreover, our framework combines both package content level and time-series level anomaly detection. Specifically, a signature database for normal behaviours of network packages is first constructed by observing regular communication patterns between field devices in a given ICS for a certain period of time. The established signature database is then incorporated into a Bloom filter to find anomalous network packages. To address the temporal dependencies between consecutive packages, we develop an additional time-series anomaly detector based on an effective deep learning technique – Long Short Term Memory (LSTM) networks [21] [22] to learn the most likely package signatures from previously seen network packages. With the help of the time-series anomaly detector, our two-level anomaly detector demonstrates highly accurate detection performance and timely detection of anomalies.

Furthermore, most currently proposed IDS lack common datasets for testing and evaluation, making it hard to compare with other frameworks. We apply our framework to a public ICS dataset [23], created from a SCADA system for a gas pipeline, for validation, and show that it significantly outperforms other existing approaches, producing the state-of-the-art detection results.

The remaining part of this paper is organized as follows. We discuss the related work including the current state of ICS-specific IDS, machine learning based IDS and other work related to this paper in Section II. Then, we outline the research problem of this paper in Section III. This is followed by the formal introduction of the Bloom filter package content level anomaly detector in Section IV and the explanation of the LSTM network-based time-series level anomaly detector in Section V. The combined framework is presented in Section VI. We then explain the ICS dataset used in this work in Section VII, with a series of experiments we have conducted in Section VIII. The last section draws the conclusion and discusses possible extensions.

II. Related Work

In this section, we discuss the existing related work to develop intrusion detection tools for ICS. Anomaly detection has been a widely applied defensive measure for decades, e.g. detecting anomalous behaviours of programs [24] [25] [26], botnet detection [27] and intrusion detection in Internet of Things [28]. Although conventional defensive measures may be adapted and strategically deployed to protect ICS from cyber attacks [29][30], there are several difficulties that hinder this tactic. The survey paper [31] provides a taxonomy and metrics for SCADA-specific intrusion detection and prevention systems. The authors discuss the difficulties and specific requirements to develop SCADA-specific IDS such as insecure-by-design components, hard real-timeliness, limited computing resources and strict availability requirement. Existing approaches have been categorized as knowledge-based approaches, behavioural-based approaches and hybrid approaches in the paper, and evaluated in terms of a set of metrics such as degree of SCADA-specific-ness, self-security, fallacy analysis, etc. The authors of [31] also provide open issues and recommendations for SCADA-specific IDS. Another related survey paper [32] focuses on IDS developed for the broader class of systems – Cyber-Physical Systems (CPS), which have been classified based on two design metrics: detection technique (i.e. knowledge-based or behaviour-based) and audit material (i.e. host-based or network-based). The paper firstly discusses the main differences between ICT and CPS intrusion detection mechanisms, and then compares existing work in terms of the two aforementioned design metrics. Particularly the authors summarise the key advantages and disadvantages of each different type of IDS, as well as their effectiveness when applying to CPS.

One of the earliest IDS for SCADA was proposed in [16]. The IDS constructs models to capture normal system behaviours. A protocol-level model for characterizing behaviours of Modbus TCP is developed and then encoded by Snort rules to detect anomalous behaviours in this paper. Due to the static topology and regular communication pattern within process control systems, the authors claim that such a model-based approach is feasible for control networks and has the potential to detect unknown attacks. To complement conventional blacklist-based approaches, which are highly effective to detect well-known attack patterns, an anomaly-based IDS is proposed in [33][17] to construct normal behaviours models over time by correlating system events in terms of their dependencies and occurrences. Particularly, the authors suggest that the proposed system provides a promising basis to combat Advanced Persistent Threat (APT) where deliberately slow-moving attack methods are normally used. The proposed IDS has been demonstrated in a small-scale pilot case under real-world conditions. A similar type of attack for ICS is addressed

in [34] by introducing a sequence-aware method to detect attacks involving a sequence of events (i.e. termed as semantic attacks).

As emphasised by both survey papers, SCADA-specific IDS need to incorporate SCADA-specific protocols and standards. The work presented in [19] provides an IDS specifically for the international standard IEC 60870-5 for ICS by using a Deep Packet Inspection technique, where signature-based rules are implemented by Snort to identify suspicious behaviours, with a complementary model-based mechanism for unknown attacks. The authors in [20] propose a stateful intrusion detection framework for smart grid systems, and demonstrate an application of such framework for IEC 61850 implemented by the open source IDS tool Suricata. The proposed method defines a set of stateful rules which are then examined with incoming network traffic. In addition to the IEC standards, the paper [18] concentrates on IDS for Modbus/DNP3 specifically. The authors suggest that network-based IDS is more suitable for SCADA than host-based IDS as they require less resources and seamlessly integrate with SCADA. A state-based IDS is proposed in this paper where a virtual image (i.e. a digital representation of the system) is setup up by predefined system knowledge. The virtual system image keeps updated by means of analysing network packages changing the physical states of the system. An alert would be raised if any packet brings the system into a critical state. The method presented in [18] has the potential to detect unknown attacks, because the alert would be triggered by critical state patterns rather than a specific attack. Based on this, the authors further conduct Critical State Analysis and State Proximity in the work [35]. Collective anomalies that are caused by a chain of seemingly licit commands can be detected by this framework.

Recently, many machine learning techniques which can automatically learn patterns from past examples and construct self-evolving models for further classification, have been applied to develop anomaly-based IDS for ICS. Specifically, these anomaly-based IDS employ available data to create normal behaviour profiles of ICS network, then detect anomalies which are deviant with the profiles, and thus are able to detect new attacks. For example, the paper [36] used one-class classification techniques which are the Support Vector Data Description (SVDD) and the Kernel Principal Component Analysis (KPCA) for intrusion detection in SCADA systems. Statistical Bayesian Networks were used in the work [37] to improve the accuracy of anomaly detection for SCADA systems. The work effectively reduced the false positive rate by combining multiple anomaly detection mechanisms such as n-grams and invariant induction. The authors in [38] applied common path mining techniques for intrusion detection of power systems. A Bloom filter based IDS for smart grid SCADA was proposed in [39] where the regular communication patterns of SCADA and the physical states of power systems have been leveraged to develop a light-weight and fast IDS for SCADA. A Bloom filter is also used in our work for a package content level anomaly detector, but we also greatly enhance our detection mechanism with an extra LSTM network-based time-series level detection model to further improve the performance. To our knowledge, this is the first time that LSTM networks have been used for anomaly detection in ICS networks. More importantly, we also show that the combination of the Bloom filter and the LSTM network model can significantly outperform the existing machine learning techniques on anomaly detection for ICS networks. Detailed performance comparison between our model and other machine learning techniques on a gas pipeline dataset can be found in Table IV.

III. Problem Statement

Anomaly detection systems for ICS are often deployed by monitoring the network traffic between field devices such as PLCs, actuators and sensors. Without loss of generality, we represent the network packages exchanged between field devices in an ICS network as a time series $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$, where each point $\mathbf{x}^{(t)}$ in the time series is an m-dimensional vector $\{x_1^{(t)}, x_2^{(t)}, \ldots, x_m^{(t)}\}$, whose elements correspond to m features that can be extracted from a package between the devices. A package level anomaly detection model will classify whether a network package $\mathbf{x}^{(t)}$ is anomalous solely depending on the features of $\mathbf{x}^{(t)}$, whilst a time-series level will conduct the classification based on the package together with a limited number of previously seen packages. In this work, we show a method which combine these two levels into a single anomaly detection framework. Moreover, both models in this work are trained only by using a time series dataset without any presence of anomalous packages, but can nevertheless be used to detect unseen anomalies.

IV. Package Level Anomaly Detection

Since, in many cases the network configuration and communication patterns between field devices in ICS are considered to be relatively stable, we assume the normal behaviour of network packages exchanged between field devices can be observed given a sufficiently large time-series dataset $\mathbf{X}^N$ without the presence of anomalies ($\mathbf{X}^N$ can be obtained by operating the ICS in "air-gapped" separation for a period of time). We capture the normal profile of network packages by establishing a signature database for the network packages in the dataset $\mathbf{X}^N$, and any packages whose signature cannot be found in the database are classified as anomalies during the detection phase.

A. Generating Signatures for Packages

All the features in network packages can be used to define signatures. We try to maximise the use of them. Specifically, the generation of package signatures involves an important step in which we transform the original feature vector $\mathbf{x}^{(t)} = \{x_1^{(t)}, x_2^{(t)}, \ldots, x_m^{(t)}\}$ of an arbitrary network package to an o-dimensional ($o \leq m$) vector $\mathbf{c}^{(t)} = \{c_1^{(t)}, c_2^{(t)}, \ldots, c_o^{(t)}\}$, in which each element $c_i^{(t)}$ is either a discrete-valued feature as the same within the original feature vector or the discretized representation of one or several continuous features (including numerical features with a large value domain) in the original feature vector. Then, the signature of a package is generated by the function:

$$s(\mathbf{x}^{(t)}) = g(c_1^{(t)}, c_2^{(t)}, \ldots, c_o^{(t)})$$

which satisfies:

$$g(c_1^{(t)}, c_2^{(t)}, \ldots, c_o^{(t)}) = g(c_1^{(t')}, c_2^{(t')}, \ldots, c_o^{(t')}) \iff c_i^{(t)} = c_i^{(t')} \quad \forall i \in (1, 2 \ldots, o)$$

Intuitively, $g(\cdot)$ is a generating function which assigns a unique value to each different combination of its parameters, and the simplest way to define $g(\cdot)$ is to concatenate the parameters to a string with a special character as the separator of the parameters.

B. The Granularity of Feature Discretization

Clearly, the functions for discretizing continuous features are of vital importance for establishing a proper signature database which captures the normal profile of network packages. Some continuous features, such as the time interval between consecutive packages sent by a field device, exhibit clustering characteristics by nature, and thus can be easily discretized. However, many other continuous features do not have natural clusters, and thus may be discretized in many different possible ways. In these cases, the granularity of discretization is a critical factor which can affect the performance of our package level anomaly detector. Specifically, if the granularity of discretization is too coarse-grained, many anomalies will be classified as normal packages (high false negative). Otherwise, many normal packages will be classified as anomalies (high false positive) if the granularity of discretization is too fine-grained. For this reason, we propose a method which can be used to find the most fine-grained granularity of discretization (which is expected to minimize the false negative rate) below an acceptable false positive rate.

Specifically, we split the dataset $\mathbf{X}^N$ into a training set $\mathbf{X}_T^N$ and a validation set $\mathbf{X}_V^N$. Let $\{n_1, n_2, \ldots, n_l\}$ be the number of discretized values for the continuous features (excluding those with natural clusters), θ be the acceptable false positive rate. Then, we establish the signature database from the training set $\mathbf{X}_T^N$ with the discretization granularity $\{n_1, n_2, \ldots, n_l\}$, and the false positive rate can be estimated by the validation error, which is proportion of packages in the validation set $\mathbf{X}_V^N$ whose signatures cannot be found in the established signature database. Therefore, we can plot the validation error denoted as err_v by the following function: $err_v = f(n_1, n_2, \ldots, n_l)$, and the optimal choice of discretization granularity can be found by

$$\underset{n_1,n_2,\ldots,n_l}{\operatorname{argmax}} \sum_{i=1}^{l} w_i n_i, \quad f(n_1, n_2, \ldots, n_l) < \theta.$$

where w_i is a weight which denotes the relative importance of the degree of discretization for the continuous feature(s).

C. The Bloom Filter Anomaly Detector

Since there can be limited memory and computing resources in ICS network traffic monitors, we use a Bloom filter to efficiently store the signature database of normal network packages and detect anomalies thereafter.

Specifically, a Bloom filter is a probabilistic data structure that is used to test whether an element is a member of a set. It consists of two components: a m-bit vector $\mathbf{v}$ in which all elements are initialized to 0, and a set of k different predefined hash functions $h_1, h_2, \ldots, h_k$, each of which maps an element to one of the m positions in $\mathbf{v}$. The insertion of an element $\mathbf{e}$ is conducted by hashing the element k times using the predefined hash functions, and then setting the positions $h_1(\mathbf{e}), h_2(\mathbf{e}), \ldots, h_k(\mathbf{e})$ in the bit vector $\mathbf{v}$ to 1. The lookup of an element $\mathbf{e}$ can be simply conducted by hashing $\mathbf{e}$ k times using the same hash functions, and then checking if all the positions $h_1(\mathbf{e}), h_2(\mathbf{e}), \ldots, h_k(\mathbf{e})$ in the bit vector $\mathbf{v}$ are 1. False positive lookup results are possible but false negatives are not. The trade-off between the false positive rate and the memory requirement can be controlled by tuning the parameters m and k.

Taking advantages of its constant lookup time and memory-efficient merits, we use a Bloom Filter as a fast and light-weighted package level anomaly detector. Specifically, let $\mathcal{B}$ be our Bloom filter, S be the set of all normal package signatures in the database, we insert all the package signatures $s \in S$ into $\mathcal{B}$ during the training phase. Thus, the package level anomaly detection function can be described as:

$$F_p(\mathbf{x}^{(t)}) = \begin{cases} 1 & \text{if } s(\mathbf{x}^{(t)}) \notin \mathcal{B} \\ 0 & \text{otherwise} \end{cases}$$

where we say $\mathbf{x}^{(t)}$ is classified as anomaly if $F_p(\mathbf{x}^{(t)}) = 1$, otherwise we say the package passed our package level anomaly detector.

V. Time-series Level Anomaly Detection

Network packages which pass the Bloom filter anomaly detector can still exhibit anomalous behaviour which can only be detected given the observation of previous packages. Therefore, we also propose a stacked LSTM neural network model for time-series level anomaly detection.

LSTM networks [21], [22], a type of recurrent neural network (RNN) with special units called 'memory cells', have been successfully applied to many sequence learning tasks such as speech recognition [40], machine translation [41] and natural language generation [42]. It has been shown that LSTM networks can also outperform traditional RNNs and feed-forward networks using fixed size time windows on numerous temporal processing tasks [43], [44]. More specifically, LSTM networks have a complex structure which allows them to "memorize" information for an extended number of timesteps, and then use the information to predict the behaviour in the next timestep. The "memory" is stored in a vector of memory cells. In each memory cell, the flow of information into and out of the cell is scaled by the learned input and output gates. The forget gates can be learned to reset memory cells to remember useful old information and discard irrelevant old information. The architecture of a memory cell with input vector $\mathbf{x}_t$, hidden input vector $\mathbf{h}_{t-1}$ (from previous timestep) and output vector $\mathbf{h}_t$ is illustrated in Figure 1. The implementation of the memory cell can be represented by the following equations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{g}_t &= \tau(\mathbf{W}_g \mathbf{x}_t + \mathbf{U}_g \mathbf{h}_{t-1} + \mathbf{b}_g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tau(\mathbf{c}_t)
\end{aligned}$$

In the above: σ is the logistic sigmoid function; τ is the cell input and output non-linear activation functions, generally using the hyperbolic tangent function; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{g}_t$, $\mathbf{c}_t$ are respectively the input gate, forget gate, output gate, cell

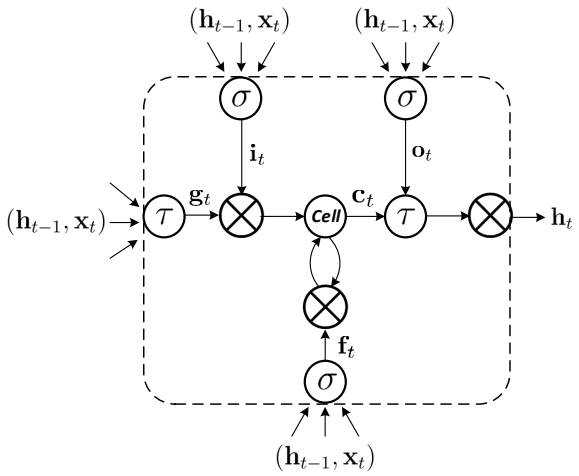

Fig. 1: The architecture of a LSTM memory cell

input activation and cell state vectors; $\mathbf{W}_\alpha$, $\mathbf{U}_\alpha$, $\mathbf{b}_\alpha$ where $\alpha \in \{i, f, o, g\}$ are the paramter matrices and vectors to be learned; and $\odot$ means element-wise product. Because of these memory cells, LSTM networks which are composed by them can obviate the need for a predefined fixed size time window, and can accurately model multivariate time-series sequences.

A. The Stacked LSTM Network-based Anomaly Detector

In general, LSTM network-based time-series anomaly detectors take the input of time-series $\{\mathbf{x}^{(t-1)}, \mathbf{x}^{(t-2)}, \ldots\}$, learn their higher dimensional feature representations, and then use those features to predict the next data point $\hat{\mathbf{x}}^{(t)}$. Furthermore, the predicted data point can be used to classify if $\mathbf{x}^{(t)}$ is anomalous by checking the similarity between $\mathbf{x}^{(t)}$ and $\hat{\mathbf{x}}^{(t)}$ [45], [46]. However, in our case, due to the high dimensionality and the coexistence of various feature types (continuous, numerical, categorical) in the network packages, it is rather difficult or even impossible to precisely reconstruct $\mathbf{x}^{(t)}$ as well as to define meaningful similarity metrics between two packages. In order to overcome these problems, our model learns to predict the signature of the next package instead of the package itself. Specifically, letting $\mathbf{c}^{(t)} = \{c_1^{(t)}, c_2^{(t)}, \ldots, c_o^{(t)}\}$ denote the discretized feature vector of a package $\mathbf{x}^{(t)}$ as described in the previous section, S be the set of all package signatures in our signature database, our model will output

$$\Pr(s \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots) \quad \forall s \in S,$$

where s is a package signature in the signature database. Furthermore, let $S^{(k)} = \{s_1, s_2, \ldots, s_k\}$ be the set which contains the top kth most probable signatures predicted by our model given $\{\mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots\}$, then, our time-series level anomaly detection function takes the form as follows:

$$F_t(\mathbf{x}^{(t)} \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots) = \begin{cases} 1 & \text{if } s(\mathbf{x}^{(t)}) \notin S^{(k)} \\ 0 & \text{otherwise.} \end{cases}$$

1) The Model Structure: Figure 2 shows the structure of our stacked LSTM network model. Specifically, the input of the model are the previous network packages with discretized feature representation (each feature is one hot encoded), the features will be passed to several LSTM layers to learn high

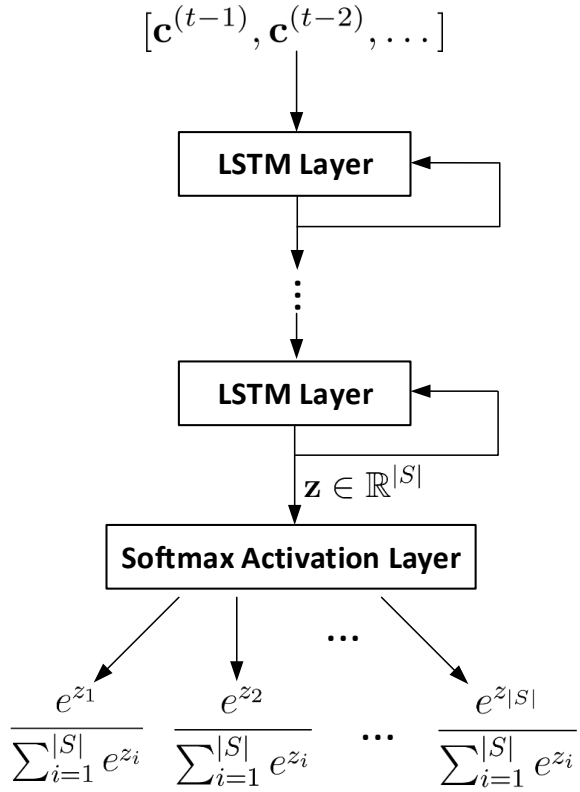

Fig. 2: The architecture of the stacked LSTM-based softmax classifier model

dimensional temporal features. The vector $\mathbf{z} \in \mathbb{R}^{|S|}$ is the output vector from the last LSTM layer, where $|S|$ is the number of unique signatures in the signature database. It is followed by a softmax activation layer which transfers the vector $\mathbf{z}$ to a $|S|$-dimensional vector of real values in the range $(0, 1)$ that add up to 1. The output layer contains $|S|$ units after the softmax activation layer, where the value in each output unit is the probability of the signature of next package being a particular value given the time-series input:

$$\Pr(s_i \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots) = \frac{e^{z_i}}{\sum_{k=1}^{|S|} e^{z_k}} \quad \forall i \in \{1, 2, \ldots, |S|\}$$

, and it is guaranteed that

$$\sum_{i=1}^{|S|} \Pr(s_i \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots) = 1.$$

The model is then trained to minimize the softmax loss function (which is also known as the multiclass cross-entropy loss and is often used as the last layer in deep neural nets for multiclass classification [47], [48]) defined as follows:

$$\mathcal{L} = -\sum_t \sum_{i=1}^{|S|} \mathbf{1}(s(\mathbf{x}^{(t)}) = s_i) \ \ln \Pr(s_i \mid \mathbf{c}^{(t-1)}, \mathbf{c}^{(t-2)}, \ldots)$$

where $\mathbf{1}(x)$ is an indicator function which yields one only if x is true, and outputs zero otherwise. The authors in [49] show that the softmax loss is indeed top-k calibrated, which means we can obtain a classifier which is guaranteed to achieve lower

top-k error with more available training data by minimizing the loss function $\mathcal{L}$. Intuitively, this means the model is well-matched with our time-series level anomaly detection function $\mathcal{F}_t$ which is based on the prediction of the top-k most likely signatures given the time-series input.

2) The Choice of k: Just like the discretization granularity of continuous features in the package level anomaly detection, the value of k in our time-series level anomaly detection function $\mathcal{F}_t$ is also a critical factor which can affect the performance of our time-series level anomaly detector. Here, we explain how to choose a reasonable value of k.

Specifically, let err_k denote the top-k error of the stacked LSTM network model on a dataset without the presence of anomalies:

$$err_k = \frac{\sum_t \mathbf{1}(s(\mathbf{x}^{(t)}) \notin S^{(k)})}{T}$$

where T is the total number of predictions made in the dataset. Since our time-series level anomaly detection function will classify packages whose signature is not included in $S^{(k)}$ as anomalies, err_k can be thought as an estimation of the false positive rate of the model when it is used for anomaly detection.

Therefore, we split the dataset $\mathbf{X}^N$ into a training set $\mathbf{X}_T^N$ and a validation set $\mathbf{X}_V^N$. We train the stacked LSTM network model using the training set. Then, the optimal choice of k can be found by:

$$\underset{k}{\text{argmin}} \quad err_k < \theta \text{ for } \mathbf{X}_V^N,$$

which means we choose the minimal k which satisfies $err_k < \theta$ on the validation set, where θ is the threshold on the estimated false positive rate when using the model for time-series level anomaly detection.

3) Adding Probabilistic Noise During Model Training: Since the dataset $\mathbf{X}^N$ does not contain any anomalous packages, the model we trained using $\mathbf{X}^N$ might be too sensitive to anomalous packages which will bring down the performance of our time-series level anomaly detector during the detection phase. More specifically, our stacked LSTM network model uses previous packages for the prediction of the signature of next package. However, if the previous packages contain one or several anomalies, the model is likely to fail in predicting the correct signature of next package. As a result, it is likely that a normal package will be classified as anomaly if the package is right after some anomalous packages. Consequently, the false positive rate of our time-series level anomaly detector will be increased unexpectedly. Therefore, in order to make our model more robust to time-series input with anomalies, we propose a strategy to train our model with artificial probabilistic noise.

Concretely, during the training phase, for each package $\mathbf{x}^{(t)}$, when it is used in the time-series input for the prediction of latter package signatures, with probability:

$$p = \frac{\lambda}{\lambda + \#(s(\mathbf{x}^{(t)}))},$$

we add some noise to the discretized feature vector $\mathbf{c}^{(t)}$, where $\#(s(\mathbf{x}^{(t)}))$ is the times of the signature $s(\mathbf{x}^{(t)})$ appearing in the training dataset, λ is a real number which reflects the expected frequency of anomalies. Note that by this probability definition, packages whose signatures with low frequencies in the training dataset are more likely to be chosen as noisy input. This is because that we expect these packages are more close to real anomalous packages, thus the model trained in this way shall be more generalised to the real situation during the detection phase.

More specifically, the noise is added by the following rule: let $\mathbf{c}^t = \{c_1^{(t)}, c_2^{(t)}, \ldots, c_o^{(t)}\}$ be the discretized feature vector of package $\mathbf{x}^t$, we first generate a random number d uniformly sampled between $[1, l]$, where $l < o$. Then, d features in $\mathbf{c}^t$ which are chosen randomly will be artificially changed to a different value. Furthermore, we add a new feature $c_{o+1}^{(t)}$ to all packages, where $c_{o+1}^{(t)}$ is set to 1 if the package contains probabilistic noise, otherwise it is set to 0, Intuitively, this additional feature can be thought as an extra bit of information to tell the model that the particular package is an anomaly or not, and thus the model should be more easily trained to be robust to noisy input. Furthermore, when the model is used during the detection phase, the additional feature of any packages classified as anomalies will be set to 1, otherwise it is set to 0. Then, the package will be used as the input for the classification of latter packages without any further processing.

VI. The Combined Anomaly Detection Framework

Having presented both our package level and time-series level anomaly detection model, we now introduce the combined framework of these two models.

Specifically, the schematic structure of the combined framework is given in Figure 3. As can be seen from the figure, the combined framework is rather straightforward. Concretely, when a package is being analysed, its signature will be firstly checked by our Bloom filter anomaly detector. If its signature is not contained in the Bloom filter, then the package will be classified as an anomaly. There is no need to pass the detected anomaly by the Bloom filter to the time-series level anomaly detector since its signature is not even in the signature database, thus it will always be classified as anomalous in the time-series level. Note that this mechanism should work well since the false positive rate of the Bloom filter detector can be estimated by the validation error during the training phase, and thus can be well controlled by tuning the granularity of feature discretization.

Furthermore, if the package passed our package level detector, then our time-series level detector will classify whether or not it is actually anomalous by checking whether or not its signature is within the predicted top k most probable signatures based on the discretized feature vectors of previous packages. Packages no matter classified as normal or anomalous, will be used as the time series input for the classification of future packages.

VII. Dataset

The dataset used in this paper for training and testing our combined anomaly detection framework was originally proposed in [23]. The network traffic data log was captured from a SCADA system for a laboratory-scale testbed of a gas pipeline system including both normal operation and real

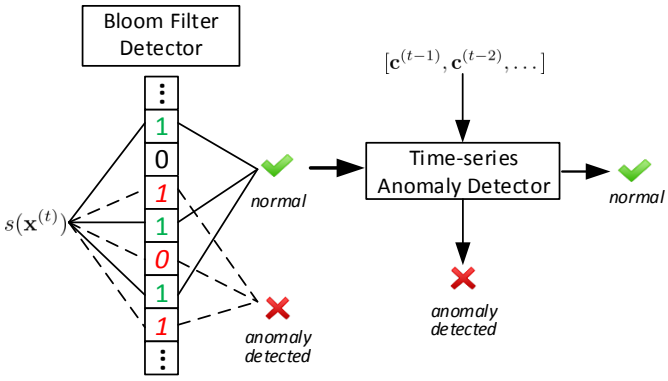

Fig. 3: The schematic structure of the combined framework for package and time-series level anomaly detection

cyber attacks. Specifically, the gas pipeline system consists of a small airtight pipeline connected to a compressor, a pressure meter and a solenoid-controlled relief valve. The system attempts to maintain the air pressure in the pipeline using a proportional integral derivative (PID) control scheme. The associated SCADA system uses the Modbus application layer protocol. An AutoIt automation and scripting language script [50] is used to initiate attacks which can inject, delay, drop and alter network traffic. Network packages are timestamped and recorded in a log file. Each network packet includes header and Modbus payload, with 20 unique features that are stored in Attribute Relationship File Format (ARFF). Table I enumerates these features in details.

TABLE I: Features in ARFF format [23]

Feature	Description
address	The station address of the Modbus slave device
crc rate	The Cyclic-Redundant Checksum rate
function	Modbus function code
length	The length of the Modbus packet
setpoint	The pressure set point for the automatic mode
gain	PID gain
reset rate	PID reset rate
deadband	PID dead band
cycle time	PID cycle time
rate	PID rate
system mode	automatic (2), manual (1) or off (0)
control scheme	Either pump (0) or solenoid (1)
pump	Pump control – open (1) or off (0) only for manual mode
solenoid	Valve control – open (1) or closed (0) only for manual mode.
pressure measurement	Pressure measurement
command response	Command (1) or response (0)
time	Time stamp

The AutoIt script randomly chooses to send legal commands or launch cyber attacks. There are four main categories of attacks considered in this dataset – command injection, response injection, denial of service and reconnaissance. These four categories are further divided into seven specific types of attacks as outlined in Table II. There are 214,580 normal network packages and 60,048 packages with attacks in total created in the dataset. More details about this dataset can be found in [23][51]. The dataset can be accessed from the webpage https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets.

TABLE II: Attack types and description in the dataset [23]

ID	Type	Description
1	NMRI	Inject random response packets
2	CMRI	Hide the real state of the controlled process
3	MSCI	Inject malicious state commands
4	MPCI	Inject malicious parameter commands
5	MFCI	Inject malicious function code commands
6	DoS	Denial of service targetting communication link
7	Recon.	Pretend of reading from devices

VIII. Experiments

In our experiments, we split the dataset into three groups according to the proportion $6 : 2 : 2$. Specifically, 60% of the data is used as our training set, 20% is used for validation set, and the remaining 20% is used for testing our anomaly detection framework. Moreover, both the training and validation set do not contain any anomalies, whereas there are anomalies distributed all over the testing set. Note that anomalous packages in the training and validation set are manually removed. After removing the anomolous packages in the training and validation set, the normal package time-series sequence is divided into many fragments. Thus, we also remove time-series fragments which are shorter than 10 packages in order to guarantee the functionality of our time-series anomaly detector.

A. Model Training and Validation

1) The Bloom Filter Anomaly Detector: In order to setup the Bloom filter which stores the signature database of normal packages for anomaly detection, we first need to discretize the continuous features in the packages of the gas pipeline dataset. Specifically, there are 9 continuous features in the packages which are the time interval between consecutive packages (calculated by the difference of the time stamp between consecutive packages), crc rate, setpoint, pressure measurement and the five PID control parameters. The five PID control parameters shall be clustered together since they are strongly correlated. Furthermore, according to the distribution of the values of the remaining four features as illustrated in Figure 4, we cluster both the time interval and crc rate into two groups using Kmeans clustering. Setpoint and pressure measurement do not have natural clusters. Therefore, in order to decide the optimal granularity of discretization, we plot the validation error (proportion of packages in the validation set treated as anomalies by our Bloom filter anomaly detector) under different granularities of feature discretization in Figure 5. Finally, the discretization strategies for all the continuous features in the gas pipeline dataset are summarised in Table III. Specifically, the values of pressure measurement and setpoint are evenly partitioned into 20 and 10 intervals respectively because we think the discretization granularity of pressure measurement is more important than setpoint, and the PID control parameters are clustered into 32 groups using

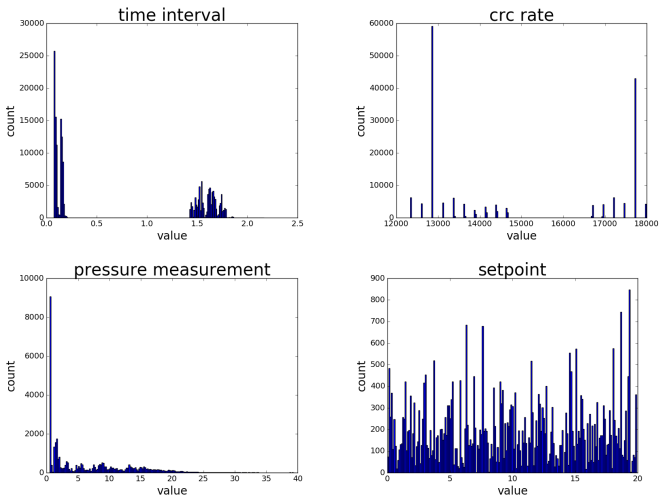

Fig. 4: The histograms for the continuous feature values each with 200 bins

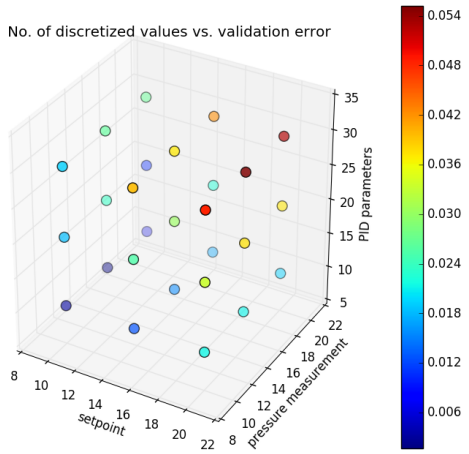

Fig. 5: The validation error under different granularities of feature discretization

Kmeans clustering. Note that we also assign an additional discrete value to each feature to represent those values that cannot be assigned to any of the clusters (e.g., if the value is more far away from any centroids than any other values in the training set) or intervals. These additional values are useful for adding probabilistic noise during the training of our time-series level anomaly detector to make it more generalised to anomalous packages with out-of-range feature values. With this discretization granularity, there are 613 unique signatures in our signature database, and the expected false positive rate of package level anomaly detection is tuned to below 0.03 as can be seen in Figure 5.

Feature	Discretization method	Value No.
time interval	Kmeans clustering	2+1
crc rate	Kmeans clustering	2+1
pressure measurement	Even interval partition	20+1
setpoint	Even interval partition	10+1
PID parameters	Kmeans clustering	32+1

TABLE III: Feature discretization strategies for the gas pipeline dataset

2) The Stacked LSTM Network-based Anomaly Detector: The model we used for time-series level anomaly detection for the gas pipeline dataset is a stacked two layer LSTM network with a structure as illustrated in Figure 2. Specifically, each LSTM layer has 256 fully connected memory units. The output layer has 613 units as expected which is equal to the size of our signature database. We train the LSTM network with and without adding probabilistic noise, both with 50 training epochs to minimize the softmax loss function. We set $\lambda = 10$ for adding probabilistic noise since the frequency of anomalies in the dataset is rather high in our experiments. However, in practice, λ should be set to a much smaller value because the frequency of anomalies should be very low in reality.

Figure 6 shows the top-k error on the training set and the validation set in both scenarios after the 50 training epochs. It can be seen that in both cases, the error ratio converges quickly to 0 with the increase of k. This means our model is rather appropriate for time-series level anomaly detection since the expected false positive rate should be quite low even with a small value of k. Moreover, with a smaller value of k, our time-series level anomaly detector should also have a lower expected false negative rate. Furthermore, the model we trained with probabilistic noise has similar top-k error with the model trained without noise (after k=3). This means the stacked LSTM network model can be actually trained to be more robust to noisy inputs which mimic the behaviour of anomalies. Lastly, according to Figure 6, by setting the acceptable false positive rate of the time-series level anomaly detector to 0.05, we can choose $k = 4$ since it is the minimal value of k with which the top-k error of the model trained with probabilistic noise on the validation set is below 0.05. More importantly, we can also observe that the top-k error on validation set drops rather steeply when $k < 4$, but it falls slowly when $k > 4$. This also indicates that $k = 4$ should be the optimal choice.

The training time of the LSTM network model for the 50 epochs which allows the the softmax loss function to converge is about 35 minutes on a Linux Machine with 3.4 GHz Intel CPUs, 15.6 GB memory size. This is rather acceptable since the training can always be conducted in a standalone, non-operational ICS mode. The average total time cost of making a classification using our combined framework is about 0.03 milliseconds on the same machine. The memory cost to store the two level anomaly detection models is 684 KB. Both should be acceptable for contemporary ICS network traffic monitors.

B. Anomaly Detection Results

We evaluate the performance of our combined anomaly detection framework on the classification results for the test set. Four metrics (precision, recall, accuracy, F1-score) which are commonly used for evaluating the effectiveness of anomaly

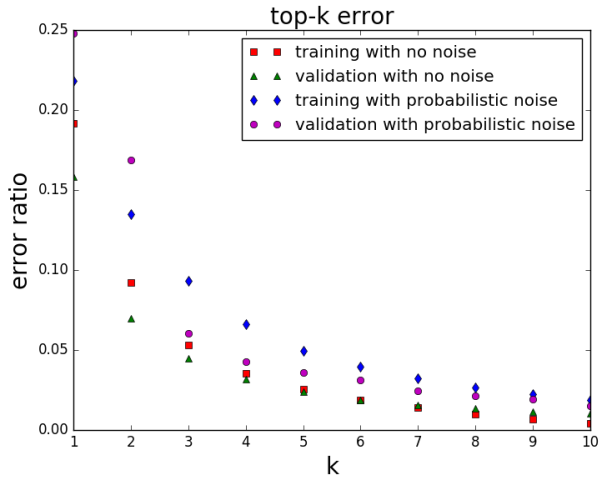

Fig. 6: The top-k error of the stacked LSTM network model on the training and validation set with and without adding probabilistic noise

detection models are analysed in our experiments. Specially, let TP denote the true positives (anomalous packages correctly identified), TN denote true negatives (normal packages correctly identified), FP denote false positives (normal packages incorrectly classified as anomalies), and FN denote false negatives (anomalous packages incorrectly classified as normal packages). The precision is then calculated as $TP/(TP+FP)$, which measures the probability of a detected anomaly is correct. The recall is given by $TP/(TP+FN)$, which measures the fraction of anomalies that are successfully identified. The accuracy is $(TP+TN)/(TP+TN+FP+FN)$, which measures the fraction of data samples that are correctly classified. Lastly, the F1-score is computed as $2\times Precision\times Recall/(Precision+Recall)$, which measures the harmonic mean of precision and recall. The F1-score is often considered as a more rounded measure of the performance for anomaly detectors.

We illustrate the results of our combined anomaly detection framework on the four metrics in Figure 7. It can be seen that by adding probabilistic noise during the training phase for the time-series level anomaly detector, the precision, accuracy and F1-score of our model can be improved especially with small values of k, this is because the trained model with probabilistic noise is less sensitive to anomalous time-series input, thus can avoid many false positives. Furthermore, we find that the stacked LSTM network is robust to anomalous time-series input even without adding probabilistic noise during the training phase since the performance of both models converges quickly with the increase of k. This means our stacked LSTM network model is robust with respect to real-wold noisy inputs (e.g., packages and sensor reading may be lost randomly) from ICS networks. The recall of both models falls quickly with the increase of k, which indicates some anomalous packages (caused by mimicry attacks) are rather close to normal packages, thus in order to detect them, a high false positive rate has to be induced. This means the choice of k plays an important role for the performance of our framework. More importantly, we also note that our choice

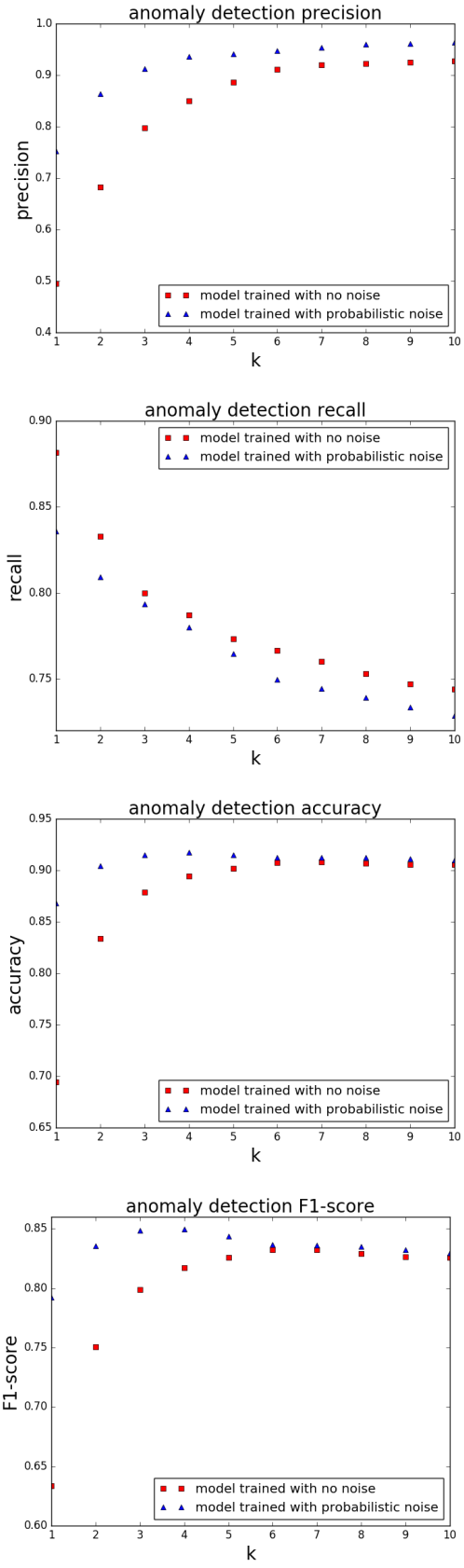

Fig. 7: The performance metrics of anomaly detection using the combined framework trained with and without probabilistic noise with different values of k

of $k = 4$, which is only chosen by observing the training and validation datasets without any anomalous packages, can achieve the highest F1-score in the detection phase, meaning the tuning of the parameters in our model is effective.

C. Comparison with other Anomaly Detection Models

In order to convincingly show the effectiveness of our combined framework for anomaly detection, we also compare its performance with other commonly used anomaly detection models for ICS.

Specifically, we have implemented four models: a Bloom Filter (BF) model; a Bayesian Network (BN) model whose structure is automatically learned from training data [53]; a Support Vector Data Description (SVDD) model [54]; and an Isolation Forest (IF) model which is often considered to be more suitable for outlier detection for hybrid data [55] for anomaly detection on the same dataset. In order to make these models also consider time-series behaviour, we combine four consecutive packages, representing a complete command response cycle in the gas pipeline dataset, as a single data sample for training and testing (thus the Bloom filter used here is different than the one we used for package level anomaly detector). Moreover, all the models are also trained using the same dataset without presence of anomalous packages.

Furthermore, we also compare the results with two unsupervised models (where training dataset contains anomalies but whether a package is normal or abnormal is not labelled) which are a Gaussian Mixture Model (GMM) and a Principal Component Analysis with Singular Value Decomposition model (PCA-SVD) that are directly taken from [52] since the authors have already used them for anomaly detection on the same gas pipeline dataset.

The detailed results are summarised in Table IV, in which the time-series anomaly detector in our combined framework is trained with probabilistic noise, and k is set to 4. For other models excluding GMM and PCA-SVD, their hyper-parameters are tuned to get best F1-score with accuracy above 0.7. It can be seen that our framework displays significantly higher performance compared to the other models. The two models exhibiting closest performance to ours are the Bayesian network model and the Bloom filter model. But, their ability in detecting anomalies is still considerably lower than ours. The other four models have relatively poor performance mainly because they are not capable of dealing with such complicated data formats.

Model	Precision	Recall	Accuracy	F1-score
Our framework	0.94	**0.78**	**0.92**	**0.85**
BF	**0.97**	0.59	0.87	0.73
BN	**0.97**	0.59	0.87	0.73
SVDD	0.95	0.21	0.76	0.34
IF	0.51	0.13	0.70	0.20
GMM	0.79	0.44	0.45	0.59
PCA-SVD	0.65	0.28	0.17	0.27

TABLE IV: Performance comparison with other anomaly detection models on the same dataset (the figures for the GMM and the PCA-SVD model are directly taken from [52], but we notice that the precision, recall and F1-score do not match for the PCA-SVD model).

Attack Type	Model	Detected Ratio
NMRI	Our framework	**0.88**
	BF	0.77
	BN	0.77
	SVDD	0.01
	IF	0.13
	GMM	0.31
	PCA-SVD	0.45
CMRI	Our framework	**0.67**
	BF	0.53
	BN	0.53
	SVDD	0.02
	IF	0.08
	GMM	0.33
	PCA-SVD	0.19
MSCI	Our framework	0.62
	BF	0.18
	BN	0.53
	SVDD	0.19
	IF	0.46
	GMM	**0.66**
	PCA-SVD	0.62
MPCI	Our framework	**0.80**
	BF	0.49
	BN	0.34
	SVDD	0.26
	IF	0.08
	GMM	0.64
	PCA-SVD	0.66
MFCI	Our framework	**1.00**
	BF	**1.00**
	BN	**1.00**
	SVDD	**1.00**
	IF	0.00
	GMM	0.32
	PCA-SVD	0.54
DOS	Our framework	**0.94**
	BF	0.93
	BN	0.93
	SVDD	0.40
	IF	0.12
	GMM	0.15
	PCA-SVD	0.58
Recon.	Our framework	**1.00**
	BF	**1.00**
	BN	**1.00**
	SVDD	**1.00**
	IF	0.12
	GMM	0.72
	PCA-SVD	0.54

TABLE V: The detected ratio (recall) of anomalous packages in each attack type

We also depict the detected ratio (recall) of anomalous packages in each attack type by all models in Table V. It can be seen that our framework has better ability in detecting anomalies in almost every scenario. GMM has a slightly better detected ratio than our framework for MSCI, but its performance in other scenarios is much worse. The most significant improvement of our framework in detecting anomalies compared with BN and BF is for MPCI, suggesting our model

for time-series level anomaly detection is much more sensitive to random parameter changes than both BN and BF.

D. Further Discussion

We also note that the detection rates of our framework for CMRI, MSCI, MPCI attack types are lower than other attack types. These three attack types are all related to the physical processes which exhibit naturally noisy behviour. As a result, some attacks may be regarded as normal behaviour since the deviation caused by these attacks can be treated as normal noise. To mitigate this problem, we believe one effective strategy is to collect more training data and increase the degree of discretization of continuous features so as to improve the sensitivity to physical-process related variable' changes during the training of our models. The other strategy is to allow the value of k for time-series level anomaly detection to be adjusted dynamically during the detection phase. This requires additional work to design mechanisms for learning optimized k given previous predictions, which is outside the scope of this work.

IX. Conclusion

In this paper, we have proposed a novel ICS-specific anomaly detection framework which coherently combines a Bloom filter package level anomaly detector and a LSTM network-based time-series level anomaly detector. Just like most deep learning models, our detection framework requires a large dataset to allow the models to be properly trained. We summarize the merits of our framework as follows: (i) it is able to learn normal behaviour solely from normal data, and thus can detect unseen attacks, (ii) it is able to deal with complicated data samples with hybrid features, (iii) the training of our models is straightforward since all the parameters (the granularity of feature discretization and the value of k for time-series prediction) can be effectively set to reasonable values by our proposed methods, (iv) it has high detection performance which is demonstrated on a real gas pipeline dataset compared with many other existing anomaly detection models.

There are several promising lines of research that could proceed. First, we are seeking ways to collect larger-scale SCADA datasets to further test our framework. We expect our framework will perform better with larger datasets. Furthermore, with larger and more complicated dataset, we plan to use more complicated deep neural networks such as convolutional LSTM networks [56] which can learn local temporal features from time-series input to improve the performance of our anomaly detection framework. Lastly, in our current work, the value of k for time-series level anomaly detection is fixed. In our future work, we will design effective approaches to adjust the value of k dynamically based on previous predictions so as to improve the performance of our time-series level anomaly detector.

Acknowledgment

The authors would like to thank anonymous reviewers for their helpful comments. Cheng Feng and Deeph Chana are supported by the EPSRC project Security by Design for Interconnected Critical Infrastructures, EP/N020138/1. Tingting Li is supported by the EPSRC project RITICS: Trustworthy Industrial Control Systems, EP/L021013/1.

References

[1] S. Adepu and A. Mathur, "An investigation into the response of a water treatment system into cyber attacks," in *IEEE Symposium on High Assurance Systems Engineering (HASE)*, 2015.

[2] S. Kriaa, M. Bouissou, F. Colin, Y. Halgand, and L. Pietre-Cambacedes, "Safety and security interactions modeling using the bdmp formalism: case study of a pipeline," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2014, pp. 326–341.

[3] A. J. Wood and B. F. Wollenberg, *Power generation, operation, and control*. John Wiley & Sons, 2012.

[4] M. P. Groover, *Automation, production systems, and computer-integrated manufacturing*. Prentice Hall Press, 2007.

[5] U. S. Department of Homeland Security. (2011) Common cybersecurity vulnerabilities in industrial control systems. "www.ics-cert.us-cert.gov/sites/default/files/documents/DHS_Common_Cybersecurity_Vulnerabilities_ICS_20110523.pdf".

[6] D. Dzung, M. Naedele, T. P. Von Hoff, and M. Crevatin, "Security for industrial communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, 2005.

[7] L. Obregon, "Secure architecture for industrial control systems," *SANS Institute InfoSec Reading Room*, 2015.

[8] D. Yang, A. Usynin, and J. W. Hines, "Anomaly-based intrusion detection for scada systems," in *5th intl. topical meeting on nuclear plant instrumentation, control and human machine interface technologies (npic&hmit 05)*. Citeseer, 2006, pp. 12–16.

[9] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security," *NIST special publication*, 2011, "http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf".

[10] ICS-CERT. (2013) Ics-cser year in review. "https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_In_Review_FY2013_Final.pdf".

[11] ——. (2014) Ics-csrt monitor september 2014 - february 2015. "www.ics-cert.us-cert.gov/monitors/ICS-MM201502".

[12] ——. (2015) Incident response activity november 2014 - december 2015. "https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Nov-Dec2015_S508C.pdf".

[13] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, 2011.

[14] R. Lee, M. Assante, and T. Connway, "ICS cyber-to-physical or process effects case study paper–german steel mill cyber attack," *Sans ICS, Dec*, 2014.

[15] ICS-CERT. (2016) Cyber-attack against Ukrainian critical infrastructure. "www.ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01".

[16] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA security scientific symposium*, vol. 46. Citeseer, 2007, pp. 1–12.

[17] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Computers & Security*, vol. 48, pp. 35–57, 2015.

[18] I. N. Fovino, A. Carcano, T. D. L. Murel, A. Trombetta, and M. Masera, "Modbus/dnp3 state-based intrusion detection system," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010, pp. 729–736.

[19] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. Wang, "Intrusion detection system for iec 60870-5-104 based scada networks," in *2013 IEEE Power & Energy Society General Meeting*. IEEE, 2013, pp. 1–5.

[20] B. Kang, K. McLaughlin, and S. Sezer, "Towards a stateful analysis framework for smart grid network intrusion detection," in *Proceedings of the 4rd International Symposium for ICS & SCADA Cyber Security Research*. British Computer Society, 2016, pp. 124–131.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[23] T. H. Morris, Z. Thornton, and I. Turnipseed, "Industrial control system simulation and data logging for intrusion detection system research," in *7th Annual Southeastern Cyber Security Summit*, 2015.

[24] K. Xu, K. Tian, D. Yao, and B. G. Ryder, "A sharper sense of self: Probabilistic reasoning of program behaviors for anomaly detection with context sensitivity," in *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on*. IEEE, 2016, pp. 467–478.

[25] X. Shu, D. Yao, and N. Ramakrishnan, "Unearthing stealthy program attacks buried in extremely long execution paths," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 401–413.

[26] K. Xu, D. D. Yao, B. G. Ryder, and K. Tian, "Probabilistic program modeling for high-precision anomaly classification," in *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*. IEEE, 2015, pp. 497–511.

[27] G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection." in *USENIX Security Symposium*, vol. 5, no. 2, 2008, pp. 139–154.

[28] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[29] A. Fielder, T. Li, and C. Hankin, "Modelling cost-effectiveness of defenses in industrial control systems," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2016, pp. 187–200.

[30] T. Li and C. Hankin, "Effective defence against zero-day exploits using bayesian networks," in *International Conference on Critical Information Infrastructures Security*. Springer, 2016.

[31] B. Zhu and S. Sastry, "Scada-specific intrusion detection/prevention systems: a survey and taxonomy," in *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*, 2010.

[32] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 55, 2014.

[33] F. Skopik, I. Friedberg, and R. Fiedler, "Dealing with advanced persistent threats in smart grid ict networks," in *Innovative Smart Grid Technologies Conference (ISGT), 2014 ieee pes*. IEEE, 2014, pp. 1–5.

[34] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM, 2015, pp. 13–24.

[35] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino, and A. Trombetta, "A multidimensional critical state analysis for detecting intrusions in scada systems," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 179–186, 2011.

[36] P. Nader, P. Honeine, and P. Beauseroy, "l_p-norms in one-class classification for intrusion detection in scada systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2308–2317, 2014.

[37] J. Bigham, D. Gamez, and N. Lu, "Safeguarding scada systems with anomaly detection," in *International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*. Springer, 2003, pp. 171–182.

[38] S. Pan, T. Morris, and U. Adhikari, "Developing a hybrid intrusion detection system using data mining for power systems," *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 3104–3113, 2015.

[39] S. Parthasarathy and D. Kundur, "Bloom filter based intrusion detection for smart grid scada," in *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*. IEEE, 2012, pp. 1–6.

[40] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[41] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[42] J. Li, M.-T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," *arXiv preprint arXiv:1506.01057*, 2015.

[43] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[44] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 3. IEEE, 2000, pp. 189–194.

[45] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[46] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*. Presses universitaires de Louvain, 2015, p. 89.

[47] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[49] M. Lapin, M. Hein, and B. Schiele, "Loss functions for top-k error: Analysis and insights," *arXiv preprint arXiv:1512.00486*, 2015.

[50] J. Brand and J. Balvanz, "Automation is a breeze with autoit," in *Proceedings of the 33rd annual ACM SIGUCCS conference on User services*. ACM, 2005, pp. 12–15.

[51] T. Morris and W. Gao, "Industrial control system traffic data sets for intrusion detection research," in *International Conference on Critical Infrastructure Protection*. Springer, 2014, pp. 65–78.

[52] S. N. Shirazi, A. Gouglidis, K. N. Syeda, S. Simpson, A. Mauthe, I. M. Stephanakis, and D. Hutchison, "Evaluation of anomaly detection techniques for scada communication resilience," in *Resilience Week (RWS), 2016*. IEEE, 2016, pp. 140–145.

[53] J. Cheng, D. Bell, and W. Liu, "Learning bayesian networks from data: an efficient approach based on information theory," *On World Wide Web at http://www. cs. ualberta. ca/˜ jcheng/bnpc. htm*, 1998.

[54] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[55] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[56] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.