ABATe: Automatic Behavioral Abstraction Technique to Detect Anomalies in Smart Cyber-Physical Systems

Sandeep Nair Narayanan, Anupam Joshi, and Ranjan Bose

Abstract—Detecting anomalies and attacks in smart cyber-physical systems are of paramount importance owing to their growing prominence in controlling critical systems. However, this is a challenging task due to the heterogeneity and variety of components of a CPS, and the complex relationships between sensed values and potential attacks or anomalies. Such complex relationships are results of physical constraints and domain norms which exist in many CPS domains. In this paper, we propose *ABATe*, an *A*utomatic *B*ehavioral *A*bstraction *Te*chnique based on neural networks for detecting anomalies in smart cyber-physical systems. Unlike traditional techniques which abstract the statistical properties of different sensor values, *ABATe* learns complex relationships between event vectors from normal operational data available in abundance with smart CPS and uses this abstracted model to detect anomalies. *ABATe* detected more than 88% of attacks in the publicly available SWaT dataset featuring data from a scaled down sewage water treatment plant with a very low false positive rate of 1%. We also evaluated our technique's ability to capture domain semantics and multi-domain adaptability using a real-world automotive dataset, as well as a synthetic dataset.

Index Terms—Anomaly Detection, Cyber-physical systems, CPS, security, attack detection

1 Introduction

Headlines like *"Casino Gets Hacked Through Its Internet-Connected Fish Tank Thermometer"* [54] and *"Webcams used to attack Reddit and Twitter recalled"*[1] are increasingly common. With rapid advancements in cognitive technologies, real-time controls systems, and industrial electronics, Cyber-Physical Systems (CPS) are being deployed in a wide variety of domains including high confidence systems like medicine, critical infrastructures, and automobiles. The potential of CPS to affect critical infrastructure and lack of security controls make them a lucrative target for attackers. Attacks on cyber-physical systems cause not just economic loss but impact the reputation, trustworthiness, and wide-scale acceptability of emerging smart infrastructure that CPS and IoT enable. Stuxnet, dubbed as the world's first digital weapon [55], is a classic attack example. It infected the PLC's (Programmable Logic Controllers) of a nuclear plant and rapidly altered the speed of its isotope enrichment centrifuge, causing premature physical damage. It took several years to even detect the real reason behind early death of those centrifuges. However, why would a PLC let centrifuges run "erroneously" when the specifics of their normal operation are well understood? This question leads to our key insight that CPS follow laws of physics and behavioral constraints specific to their domain of operation. In particular, a centrifuge shouldn't have spun unnaturally fast or slowed down abnormally!

One approach to secure such systems is to add advanced security features. Due to a variety of constraints [43] including physical environment feedback, distributed control, real-time response, wide-scale geographic distribution, and multi-tiered characteristics, layering on a complex security system is impossible in many cases. The area, power, and cost requirements also put constraints on the cyber-physical system's design and make it hard to have advanced software and hardware security stack on them. For example, the geographic distribution and remote locations of the components of a power grid CPS can result in reduced network connectivity. Mobile CPS like health and fitness monitoring systems can be power constrained. Support for legacy systems is another constraint for adding such advanced security features. For example, CAN bus, a broadcast based legacy communication channel lacks even the basic authentication and authorization techniques. This deficiency is utilized in many attacks on cars which range from simple driver distractions to complete remote takeover of an unaltered car [22], [39], [51]. However, moving away from CAN networks is often not permitted due to cost and compatibility constraints.

Detect and mitigate is an alternate strategy to secure such systems. In this paper, we propose a new technique, *ABATe* (**A**utomatic **B**ehavior **A**bstraction **Te**chnique) to detect anomalous behaviors and attacks in smart cyber-physical systems. A key insight on cyber-physical systems is that their operations are governed by the laws of physics and domain norms. For example, in the automotive domain, a car (3400 Lbs with 250hp engine) will take several seconds

- *Sandeep Nair Narayanan was with the Department of Computer Science & Electrical Engineering, University of Maryland Baltimore County. Email: sand7@umbc.edu*
- *Anupam Joshi was with the Department of Computer Science & Electrical Engineering, University of Maryland Baltimore County. Email: joshi@umbc.edu*
- *Ranjan Bose was with the Department of Electrical Engineering, Indian Institute of Technology Delhi. Email: rbose@ee.iitd.ac.in*

Manuscript received Dec 19, 2018; revised Oct 11, 2020.

1. http://www.bbc.com/news/technology-37750798

to accelerate from 0 to 100 mph. Similarly, its ignition state will be "ON" when it is in motion. Such constrained normal behaviors of CPS are well represented in their respective operational data and are available in abundance. *ABATe* automatically abstracts these normal operational behaviors into context vectors by ingesting the operational data from them.

Attacks on CPS, often defined as *"intentional actions trying to cause undesired effects in the physical world"* [19], are just perceived deviations from these normal behaviors. Hence, *ABATe* looks for such deviations in the working data and identifies different attacks on the cyber-physical system. Our technique works in two stages; a domain independent "offline learning phase" to abstract the norms of the domain and an "online monitoring" phase to detect attacks by measuring their deviation from normal. An important characteristic of *ABATe* is its multi-domain adaptability (using the same technique on data from different CPS domains). We also don't need examples of attack/anomalous states – putting a CPS in "bad" states is often impossible due to safety constraints, or can cause physical damage to the system.

In this paper, we demonstrate *ABATe*'s capabilities using real-world datasets from two different CPS domains. The first dataset(SWaT) [17] is from a scaled down sewage water treatment plant where data is collected over a period of 11 days. We use *ABATe* to detect the 36 well annotated attack scenarios in this dataset. To prove *ABATe*'s multi-domain adaptability, we collected and processed more than 25 hours of real driving data from a modern car. We also use the automotive dataset to study the capability of *ABATe* to abstract context from real-world cyber-physical systems which enables it to detect attacks. Our evaluations show that *ABATe* can detect a large variety of attack scenarios with a low false positives rate. Our major contributions in this paper include the following.

1) Developed a domain-independent technique, *ABATe* to learn the normal working behavior of cyber-physical systems using sensed normal data and neural networks.
2) Used the learned model from *ABATe* to generate an $ABATe_{score}$ to detect attacks and anomalies.
3) Evaluated *ABATe* using a popular public SWaT dataset and detected various labeled attacks in it.
4) Collected, aggregated, and analyzed data from a car and used it to evaluate the context abstraction and multi-domain adaptability of *ABATe*.

The rest of this paper is organized as follows. A brief survey of related work and our attack model can be found in Section 2 and Section 3 respectively. *ABATe*'s methodology is described in Section 4 followed by a brief description of its implementation in Section 5. A detailed evaluation of *ABATe* using two real-world datasets; a SWaT dataset and a car dataset ensue in Section 6 and the paper concludes with a discussion of future work in Section 7.

2 Literature Review

Anomaly detection is successfully used in computer security [30] , biological domain [37] , mechanical domain [8], [9] , financial domain [10] , unusual activity detection [27] , user behavior [3], etc. In an extensive survey, Chandola et al. [11] define different types of anomalies which include point anomalies, contextual anomalies, etc. and a wide variety of techniques based on classification, clustering, statistics, information theory etc. to detect them.

Warrender et al. [53] proposed T-STIDE (Threshold Time-Delay Embedding) which employs a window-based approach for anomaly detection in which a database of normal subsequences is built and are compared against the test sequences. Intrusion detection methods based on HMM (Hidden Markov Model) have been suggested multiple times [16], [47]. Another anomaly detection technique based on hierarchical HMM's is proposed by Zhang et al. [56]. However, Warrender et al. [53] reported significant performance overheads when HMM based techniques are compared with other techniques in longer sequences. Keogh et al. [25] invented SAX (Symbolic Aggregate Approximation) to determine time-series discords which could be used in a wide variety of domains like telemetry monitoring, medicine, and surveillance.

Recently, neural networks got traction and several techniques have been employed. Sabokruou et al. [48] used fully convolutional neural networks to detect abnormal regions in a video. O'shea et al. [44] applied recurrent neural network for anomaly detection on air radio networks. Laskov et al. [32] developed a technique to visualize anomaly detection. It enables experts to interpret predictions made by the learning technique used. Some of the latest anomaly detection techniques include Netflix RPCA (Robust Principle Component Analysis), Yahoo EGADS [31], EXPOSE [49], HTM [2], etc. Black box analysis techniques [13] are also used to detect anomalies in large-scale systems, using externally available information.

2.1 Anomaly Detection in Cyber-Physical Systems

Detecting attacks in cyber-physical systems is a challenging task. In the CPS domain, Gollmann et al. [20] described the potential stages of a CPS and differentiates cyber-attacks from cyber-physical attacks. A survey of various anomaly detection techniques used in cyber-physical systems from Mitchell et al. [40] describes the challenges and various techniques proposed. Jones et al. [24] proposed a formal method for anomaly detection using Standard Temporal Logic (STL). If the attacks against the system are well understood, their technique can be used to infer human readable formula. Krotofil et al. [29] used statistical techniques to detect attacks on cyber-physical systems. They used information theoretic measures like sensor specific entropy and plant-wide entropy to detect attacks. Several related research focused on securing specific CPS domains. For example, research from Sridhar et al. [50], Li et al. [33], Hong et al. [21] etc. focused on securing the smart grid infrastructure while research from Bezemskij et al. [4], [5], Vuong et al. [52], etc. focused on securing robotic vehicles. In our research, we come up with a domain independent solution to secure CPS domain.

A data-driven approach is proposed by Liu et al. [36]. They use spatio-temporal features and learn the system-wide patterns using a Restricted Boltzmann Machine (RBM). Goh et al. [18] used recurrent neural networks to detect

anomalies in cyber-physical systems. They used an LSTM (Long Short-Term Memory) to model complex temporal sequences and predicted the next expected output. The deviation of the actual sensor data and the predicted data using CUSUM (Cumulative Sum Control Chart) is then used to detect attacks. However, they were only able to apply their technique on phase 1 of the SWaT dataset [17] we use for the evaluation of *ABATe*.

Several works in the power systems CPS use state estimation [14], [45] to detect attacks. DAD from Adepu et al. [1] is another distributed attack detection technique for a water treatment plant using invariants. Invariants are mathematical relationships between different properties in the system. From a CPS design, DAD extracts invariants using state entanglement, component model, state condition graphs, and state bounds. These invariants are later used to detect attacks. DAD assumes CPS as a white box system where design, environment variables, and their dependencies are easily available. Moreover, the reported invariants were extracted manually in their study. Mitchell et al. [41] introduced another technique to transform behavior rules to state machines for safety-critical systems like medical cyber-physical systems. A key insight from their work is that the accuracy of their technique is dependent on the completeness of the behavior rule set.

Many of these existing techniques use statistical properties of sensor values to detect anomalies. However, they fail to capture domain behaviors specific to the current domain and current deployment environment. In the case of cyber-physical systems, the same CPS deployed in different environment settings may behave differently. Hence, it is important to devise techniques specific to the current domain. However, the performance of techniques entirely based on the abstraction of manually crafted behaviors is directly dependent on the completeness of the domain behaviors [41]. For large industrial cyber-physical systems, creating such a complete set is an arduous task and error-prone.

Our technique *ABATe* can automatically learn general constrained behavior of CPS environments. It is capable of abstracting the domain's normal behavior in the form of vectors using its operational data and makes *ABATe* suitable for several CPS domains and deployment environments. Unlike many techniques in the literature which are evaluated against synthetic dataset's, we use two real-world datasets to evaluate *ABATe*. As noted by Goh et al. [18], many of the existing techniques are signature based where anomalous behaviors are explicitly mentioned. Hence, a direct comparison of such techniques to ours is difficult. In section 6, we evaluate our technique using a standard publicly available dataset SWaT and discuss the semantics on how our technique detects these anomalies using another automotive dataset. The section also compares results from recent research that used SWaT dataset like CNN [28], DNN [23], GAN [34], etc. It should be noted that a direct comparison to some other recent unsupervised techniques using the same dataset is not viable because some of them [18] are restricted to specific phases of the SWaT dataset, or they use domain specific characteristics [1] in their techniques.

3 Attack Model

Several research [26] focus on detecting attacks and intrusions to the network infrastructure. CPS also share similar network architectures. Hence, in this paper we focus on those attacks that come past these network defense infrastructures and affects their physical state. Our system should be used in conjunction with existing network defense architectures to protect the CPS infrastructure against common network attacks like eavesdropping, unauthorized access, etc.

Our proposed technique tries to detect anomalies based on the data used for training. Not all attacks detected by our technique are attacks. Sometimes, anomalies can be valid but rarely occurring events. In cases where the working conditions of the CPS significantly changes, our technique can be learned additively to accommodate them. Moreover, we assume that we have adequate normal data that is representative of the actual working behavior of a cyber-physical system.

4 Methodology

The operations of a cyber-physical system are constrained by the laws of physics and implicit domain behaviors. The dependencies arising from such constraints are well represented in their normal working behavior. In a rule-based system, an expert will list out all these dependencies for detecting various anomalous behaviors. It is tedious and error-prone in complex systems with tens or hundreds of different components. In this paper, we develop *ABATe* which learns these dependencies from the normal operational data and use them to detect anomalous behaviors. Our technique works in two phases; an off-line "Learning phase" and an on-line "Monitoring phase" as shown in Figure 1. The off-line Learning phase abstracts a perceived normal domain-model into context vectors from the existing data. In the on-line Monitoring phase, the live data from all the components is processed and is checked for anomalies using the generated model.

4.1 Off-line Learning Phase

The off-line phase in *ABATe* ingests the data from a cyber-physical system and abstracts its normal operational behavior into context vectors. This phase constitutes 2 major tasks; state vector generation and context vector generation.

4.1.1 State Vector Generation

Let X (Eq:1) be the set of n components in a CPS and each $x_i \ 2 \ X$ takes value $v_{i;t}$ at time t. $v_{i;t}$ can either be continuous or discrete. The raw state of a cyber-physical system at time t, $r\forall_t$ (Eq:2), is defined as a vector of the aggregation of values from all its individual components at that time instant. Let RV_{seq} (Eq: 3) be the input sequence of raw vectors. Each $r\forall_t$ is a point in an n-dimensional vector space with each $v_{i;t}$ as a specific dimension. The potential continuous nature of the sensor values causes the state space to have an infinite number of states.

$$X = fx_1; x_2; :::; x_i; :::; x_ng \quad (1)$$

$$r\forall_t = hv_{1;t}; v_{2;t}; :::; v_{i;t}; :::; v_{n;t}i \quad (2)$$

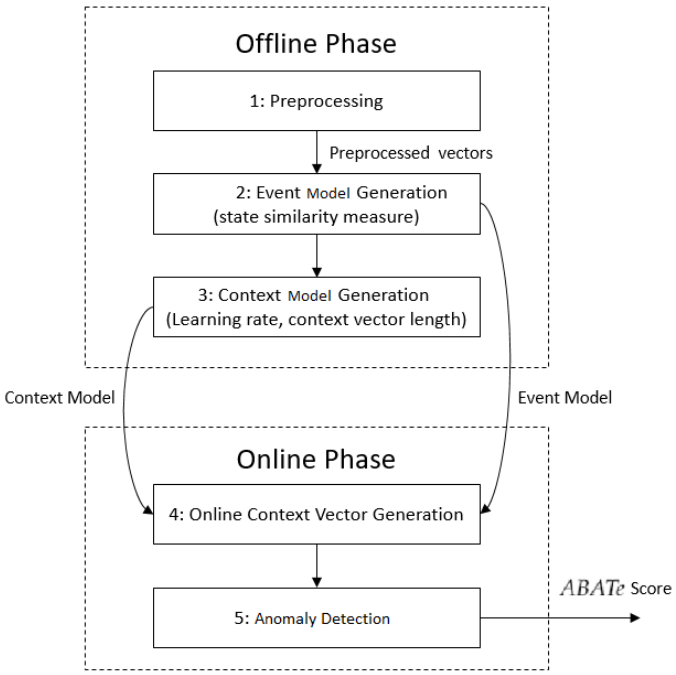

Fig. 1: *ABATe* Implementation Pipeline

$$RV_{seq} = rv_1; rv_2; :::; rv_i; :::; rv_t$$
$$where;$$
$$rv_t : raw\ vector\ at\ time\ t \tag{3}$$

We perform two optimizations on the normal operational data to generate a state vector set S, a subset of this vector space. First, we assume that "given a long enough duration of time, we can capture most normal behaviors in a cyber-physical system". It flows from the insight that CPS have constrained behavior. Hence, we add only those rv_t's which we have observed during the normal operation of the CPS. Second, we aggregate those states which are very close to each other in the state vector space and add only their centroids to S because such states often exhibit similar behaviors. In *ABATe*, we find the Euclidean distance between two rv_t's and fold them as the same state, if the distance falls below an empirically found state transition threshold, $_{sim}$. Each state in S now corresponds to a legal state in the respective CPS.

We label each state $s_i \ 2 \ S$ randomly and generate one-hot vectors [46] for each of them. In one-hot vectors, we generate d separate variables for d distinct observations of a variable. The generated hot vector set HS has a $\hat{hv}_i$ corresponding to each $s_i \ 2 \ S$. We generate the hot vectors to make sure that each $\hat{hv}_i$ is independent of each other and assume no relationship between each other based on the raw sensor values. *ABATe* will now automatically learn the complex interdependencies from the normal operational data and generate context vectors in the Context Vector Generation task.

4.1.2 Context Vector Generation

In this task, *ABATe* learns context vectors that abstract the perceived normal behavior of the CPS for each state $\hat{hv}_i \ 2 \ HS$. To learn context vectors, *ABATe* finds an embedding for each state $\hat{hv}_t$ by capturing their distributional behavior in the normal operational data. In mathematics, *"an embedding of one topological space X in another space Y is nothing more than a homeomorphism of X onto a subspace of Y. The domain X is called the embedded space and the target Y is called the ambient space"* [15]. Neural networks are successfully used to generate such embeddings in domains like Natural Language Processing(NLP) [38] that learns the distributional behavior of words in natural language text. In a similar way, *ABATe* learns the embeddings for each state vector, $\hat{hv}_i \ 2 \ HS$ using neural networks and CPS normal data. These embeddings encapsulate the distributional behavior and encode the contextual information with them. For example, let $e_i; e_j$ be the learned embeddings corresponding to $\hat{hv}_i; \hat{hv}_j$. Then, the vector similarity between e_i & e_j will be high if $\hat{hv}_i$ & $\hat{hv}_j$ has the same context (occur together frequently in the normal operational data). Conversely, the vector similarity will be minimum, if they never occur together. In the on-line monitoring phase, we utilize this property to detect context anomalies.

The neural network which learns the embedding in *ABATe* determines a function which converts $\hat{hv}_i$ into e_i as presented in Eq: 4 and Eq: 5. The network has an input layer with size equal to the number of states in HS ($HS_{size} = j \ HS \ j$). They are connected to a fully-connected neural layer with input size as HS_{size} and output size as the context vector length, C_{size} (a hyper-parameter). There will be no activation function for this layer and its output is directly used as context vectors. The final layer is a soft-max activation applied on a fully connected neural layer with input size as the context vector length (C_{size}) and output size as HS_{size}. The soft-max activation function implements the soft-max function [6] as described in equation 5. Once training is done, the first layer will act as a lookup function and convert the state vector s_i to context vector e_i. The e_i is a condensed representation of $\hat{hv}_i$ with encoded context or their distribution in the normal data. In addition, it also learns implicit relationships between states based on their contexts.

Input Layer:

$$e_i = f(\hat{hv}_i)$$
$$f(x) = Wx + B$$
$$where;$$
$$\hat{hv}_i: Hot\ Vector\ corresponding\ to\ s_i$$
$$W: Matrix\ with\ size\ HS_{size} \quad C_{size}$$
$$B: vector\ with\ size\ C_{size}$$
$$e_i: Context\ Vector\ corresponding\ to\ s_i \tag{4}$$

Output Layer:

$$Output = SoftmaxAct(f^0(c_i))$$
$$f^0(x) = W^0x + B^0$$
$$SoftmaxAct(x) = \frac{e^{x_i}}{\sum_{k=1}^{x_{size}} e^{x_k}}$$
$$where;$$
$$e_i : Context\ Vector\ corresponding\ to\ s_i$$
$$W: Matrix\ with\ size\ C_{size} \quad HS_{size}$$
$$B: vector\ with\ size\ HS_{size} \tag{5}$$

To train the network, we introduce a new variable $c\hat{hv}_i$ corresponding to each $\hat{hv}_i$. Each $c\hat{hv}_i$ is a hot vector and assumes no relationship between any existing $\hat{hv}_i$ and is

used only for training the neural network. We also introduce a context window of size cw_size while training. It defines the influence of a state to its preceding states. For example, if $cw\_size = 2$, it implies that the current state has some dependency on the previous 2 states. We show that tuning this variable adds robustness to *ABATe*.

The training set for the neural network, $TrainSet$, is a set of tuples in the format $\langle input; output \rangle$. We generate these tuples from the sequence of normal operational data of a cyber-physical system. Let $\langle \vec{hv}_1; \vec{hv}_2; :::; \vec{hv}_i; \vec{hv}_{i+1}; ::: \rangle$ be a sequence which appears in the normal operation of the CPS. Now for each $\vec{hv}_i$, we generate tuples $\langle \{\vec{hv}_{i-k}\}_{k=0}^{k=cw\_size}; \vec{chv}_i \rangle$. For example, we generate tuples $\langle \vec{hv}_{i-2}; \vec{chv}_i \rangle; \langle \vec{hv}_{i-1}; \vec{chv}_i \rangle$ and $\langle \vec{hv}_i; \vec{chv}_i \rangle$ to $TrainSet$ if $cw\_size = 2$. On training, the network will maximize the log probability of $\vec{chv}_i$, given $\{\vec{hv}_{i-k}\}_{k=0}^{k=cw\_size}$. The intuition is that the neural network will bring all the context vectors $\vec{hv}_i$ which appear inside the $context\_window$ closer, while it pushes apart those vectors which do not appear in the context window. Such behavior helps encode the normal working behavior of the cyber-physical system under consideration into context vectors. After training, a context vector c_i will be generated for each $hv_i \in HS$. These off-line stage operations are presented in algorithm 1.

Algorithm 1 Off-line Learning Phase

Input:
$RV_{seq} = \langle \vec{rv}_1; \vec{rv}_2; :::; \vec{rv}_i; ::: \rangle$
Output:
StateMdl: Raw vector to State vector mapping
CtxMdl: State vector to Context vector mapping

1: **procedure** LEARNING PHASE(RS)
 # State Vector Generation
2: $S \leftarrow FoldSimilarVectorsToOne(RV_{seq}; \theta_{sim})$
3: $HS \leftarrow GenerateHotVector(S)$
 # Context Vector Generation
4: $TrainSet \leftarrow []$
5: **for each** window $\vec{rv}_{i-cw\_size} ! \vec{rv}_i$ **in** RV_{seq} **do**
6: Append $\langle \{\vec{hv}_{i-k}\}_{k=0}^{k=cw\_size}; \vec{thv}_i \rangle$ to $TrainSet$
7: $CtxMdl \leftarrow TrainContextVectors(HS; TrainSet)$
8: $\vec{c}_i \leftarrow GetContextVector(CtxMdl; s_i)$

4.2 On-line Monitoring Phase

In the off-line phase, we created context vectors $\vec{c}_i$ corresponding to all state hot vectors $\vec{hv}_i \in HS$ and encapsulated the normal behavior of the cyber-physical system. The on-line monitoring phase aggregates live inputs from different components in the system and generates an $ABATe_{score}$ that helps detect abnormal or anomalous states in the system. Two kinds of anomalies are detected in this phase; point anomalies based on the hot-vector set HS and context anomalies based on context vector set CS.

The first step in this phase is the aggregation and generation of raw vectors, $InputRV_{seq} = \langle \vec{rv}_1^0; \vec{rv}_2^0; \vec{rv}_3^0; :::; \vec{rv}_t^0; ::: \rangle$ using inputs from different components in the CPS, where $\vec{rv}_t^0$ is the input vector at time t. As described in section 4.1.1, each $\vec{rv}_t^0$ will be a point the same n-dimensional space. We choose a state $s_i^0 \in S$ corresponding to each $\vec{rv}_t^0$ such that the Euclidean distance between s_i and $\vec{rv}_t^0$ is minimum. Now a sequence of states $InputState_{seq} = \langle s_1^0; s_2^0; :::s_t^0; ::: \rangle$ will be generated where s_t^0 is the chosen state corresponding to each $\vec{rv}_t^0$. Let EU_t be the Euclidean distance between a raw vector $\vec{rv}_t^0$ and $\vec{rv}_i$ corresponding to s_i^0. In all normal cases, EU_t should be a very small value because we assume that in a constrained system like a CPS, we have already observed most of the normal states. A small value for EU_t indicates an already observed state and higher values indicate unobserved states. Hence this value is an indicator for detecting point anomalies.

However, many anomalies that are contextual anomalies cannot be detected using this score. Contextual anomalies are those in which two already observed, but out of context states come together. As described in section 4.1.2, the context vectors encapsulate the context of each vector and hence they are used to detect such anomalies. Let $InputCV_{seq} = \langle \vec{c}_1^0; \vec{c}_2^0; :::\vec{c}_t^0; ::: \rangle$ be the context vector sequence corresponding to $InputState_{seq}$, where $\vec{c}_t^0$ is the context vector corresponding to state s_t^0. In *ABATe*, we use the cosine distance (Eqn: 6) of the current context vector $\vec{c}_t^0$ and the previous context vector $\vec{c}_{t-1}^0$ to detect context anomalies and is denoted by $ContextSim_t$. The $ContextSim_t$ determines the appropriateness of a vector to its context or nearby vectors. A lower score implies that the current vector is very appropriate in the existing context as observed from the perceived normal data used for generating the context vectors and vice versa.

It can be observed that EU_t and $ContextSim_t$ are related. When the value of EU_t is high, the confidence in that state is very low and the corresponding $ContextSim_t$ score also should be considered with lower confidence. Hence to generate a combined score to detect point anomalies and context anomalies, we use a combined score as described in Eqn: 7. The first component in the $ABATe_{score}$ corresponds to the dissimilarity of current raw vector $\vec{rv}_t^0$ to the known states. Since we use the normalized values in the raw vectors, the maximum distance possible between any two vectors in the vector space is $\sqrt{n}$, where n is the number of components in the CPS. We take the exponential because we need to magnify the fact that a higher value of EU_t increases the likelihood of point and contextual anomalies. The second component indicates the contextual appropriateness of the new state. A low value for $ABATe_{score}$ indicates that the current state is very similar to a state we have already observed and that state is contextually appropriate. On the other hand, a high value can indicate 2 situations; an observed state with low contextual appropriateness or an unobserved state, both of which indicates a deviation from the perceived normal. Hence we use this $ABATe_{score}$ for discerning anomalies by choosing an empirically estimated detection threshold $\theta_{anomaly}$. The operations of this stage are described in algorithm 2.

$$ContextSim_t = 1 - \frac{\vec{c}_t^0 \cdot \vec{c}_{t-1}^0}{\|\vec{c}_t^0\| \|\vec{c}_{t-1}^0\|} \quad (6)$$

where;
$\vec{c}_t^0; \vec{c}_{t-1}^0$: *Context Vectors corresponding to* s_t^0 *and* s_{t-1}^0

$$ABATe_{score} = e^{(EU_t)} \quad ContextSim_t$$
$$\text{where;}$$
$$ContextSim : \text{Contextual similarity from Eqn: 6}$$
$$EU_t : \text{Euclidean distance of } rv_t^0 \text{ to } s_t^0 \tag{7}$$

Algorithm 2 On-line Monitoring Phase

Input:
$InputRV_{seq}$ $r\tilde{v}_1^0; r\tilde{v}_2^0; :::; r\tilde{v}_i^0; :::$
StateMdl: Raw vector to State vector mapping
CtxMdl: State vector to Context vector mapping
Output:
$ABATe_{Score}$

```
1: procedure MONITORINGPHASE(InputRV)
2:   for each rv_i^0 2 InputRV_seq do
3:     s_t^0; EU_t  getSim(StateMdl; rs̃_t^0)
4:     ContextSim_t  Eqn 6(CtxMdl; s_t^0; s_t^0 1)
5:     ABATe_score  Eqn 7(n; EU_t; ContextSim_t)
6:     if ABATE_Score > anomaly then
7:       report Anomaly
8:     else
9:       report Normal
```

5 ABATE IMPLEMENTATION

Our implementation of *ABATe* consists of a pipeline for ingesting input data, generating corresponding context vectors, and detecting on-line anomalies as shown in Figure 1. It has five stages in total which include the off-line and on-line phases as enumerated below.

1) **Preprocessing Stage:** The input from sensors come in different formats (in cars, we collected in JSON format, while the SWaT dataset is in csv format). They are converted to a common vector format after scaling the sensor values into a 0 1 scale in this stage.
2) **State Model Generation Stage:** The second stage corresponds to the generation of a state model by aggregating states which have similar properties. In *ABATe*, we consider each raw vector as a point in an n-dimensional space and use Euclidean distances between these two vectors for state aggregation. Two raw vectors are aggregated if the Euclidean distance between them falls below a certain similarity threshold, $_{sim}$ (a hyper-parameter). Once similar states are aggregated, they are labeled randomly.
3) **Context Model Generation Stage:** The context vectors are generated using the neural network as described in section 4.1.2. We implemented the neural network using Tensorflow as the backend and trained the network using cross-entropy loss function with Adams optimizer. The hyperparameters used in this step are the learning rate and context vector length.
4) **On-line Context Vector Generation Stage:** This stage takes context model, state model, and live inputs from sensors as input. After preprocessing, the live sensor inputs are used to generate a test vector rv_t^0. Then a state label s_t^0 2 S is determined from the event model by choosing the state having the closest Euclidean distance from rv_t^0. The vector e_t^0 corresponding to s_t^0 is also retrieved from the context model. The process is repeated for all the live inputs to create a stream of state vectors and context vectors. This stream of state vectors, context vectors, and Euclidean distances (EU_t) are input to the Anomaly Detection stage.
5) **Anomaly Detection Stage:** Both point anomalies and context anomalies are detected in this phase. We generate an $ABATe_{score}$ for this purpose. The $ContextSim_t$ is generated from context vectors using Eqn: 6 and it is then used in Eqn: 7 to generate the stream of $ABATe_{score}$'s. A high value for $ABATe_{score}$ denotes a potential deviation from the perceived normal while a lower score denotes normal operations.
Several single state-based or window-based techniques can be used to determine anomalies. We explored different techniques like window-average, window-median, percent change, and so forth on the sequence of $ABATe_{score}$'s. Their evaluation can be found in section 6. The simple strategy is to use a hyper-parameter $_{anomaly}$. If the calculated $ABATe_{score}$ is above this threshold, an anomaly is detected while any value below the score will be deemed as normal. The value of $_{anomaly}$ needs to be empirically estimated considering how critical the CPS setup is. If the CPS is critical, a conservative low value of $_{anomaly}$ should be chosen while non-critical systems can afford higher thresholds. We found that the average based techniques yielded best performance.

6 EVALUATION

ABATe abstracts the normal behavior of any cyber-physical system in the form of context vectors and use it for detecting deviations from normal that include attacks. In this section, we evaluate various capabilities of *ABATe*. Detecting attacks on real-world datasets, and its ability to abstract domain behaviors are two critical features of *ABATe*. We evaluate the performance of *ABATe* using 2 real-world datasets; SWaT [17] (Sewage Water Treatment) dataset and an automotive dataset. Detecting well-annotated attacks from the SWaT dataset demonstrates the ability of *ABATe* to detect attacks in real-world systems. *ABATe*'s multi-domain adaptability and ability to abstract context are demonstrated using a new automotive dataset which features almost 25 hours (or 1000 miles) of real driving data from a modern car. Finally, we demonstrate the usefulness of *ABATe*'s context window using a synthetic dataset. Our evaluations show that *ABATe* copes well with real-world cyber-physical systems to detect anomalies.

6.1 SWaT dataset Evaluation

The Secure Water Treatment (SWaT) dataset from Goh et al. [17] is collected from a fully operational scaled down

sewage water treatment plant. The six stages of the plant include

1) "RAW water Supply and storage" stage
2) "Pre-treatment" stage
3) "Ultrafiltration and backwash" stage
4) "De-Chlorination" stage
5) "Reverse Osmosis (RO)" stage
6) "RO Permeate Transfer, UF Backwash and Cleaning" stage.

The complete dataset contains readings from 51 different components some of which are discrete while some others are continuous in nature. The different components; sensors, actuators, and PLC's (Programmable Logic Control), communicate via wired or wireless interfaces and data is collected from all the 51 components every second.

The dataset has data from the plant for 11 days in total with seven days of CPS normal working behavior. The plant was put into various single-stage and multi-stage attacks for the remaining 4 days. A total of 36 different attacks were performed during this time. In many of the attacks, attackers manipulate the data received to other components which force the receiving component to behave erroneously.

The dataset contains 4 kinds of attacks. The first kind is Single Stage Single Point (SSSP) where the attack focuses on a single point in the same stage. For example, an attack is to make the PLC controlling a valve to believe that the water level in a tank is low. As a result, it will not automatically open and cause overflowing. The second type is Single Stage Multi Point where multiple components are involved in the attack. A sample attack example from the dataset is when two pumps which pump out water from a storage in the same stage are attacked together, eventually resulting in water overflow. Multi-Stage Single Point and Multi-Stage Multi-Point are the similar counterparts in which multiple stages are involved. The normal working data and the attacks in the attack datasets are properly annotated. Apart from data from its components, the dataset also contains network data while the plant is running. In this work, however, we use the measurements from its components only because, as mentioned in our attack model, we try to detect attacks from the physical characteristics of a smart cyber-physical system irrespective of the underlying network architectures or protocols.

6.1.1 Test Setup

The main objective of evaluations with the SWaT dataset is to determine the ability of *ABATe* to detect attacks in real-world cyber-physical systems. We also use this dataset to analyze the sensitivity of similarity threshold parameter on attack detection. As discussed in section 5, we can use several techniques to detect anomalies using the sequence of generated $ABATe_{score}$'s. We analyze the performance of these metrics also using the SWaT dataset. This dataset has 7 days of data about its normal operation, and then 4 days of data when attacks were performed.

The first step in *ABATe* is to generate a perceived normal behavioral model. As training data, we used the data from the first 7 days in the dataset (off-line Learning phase). The values from different sensors had different ranges. For example, some of them are binary (eg. valve status;

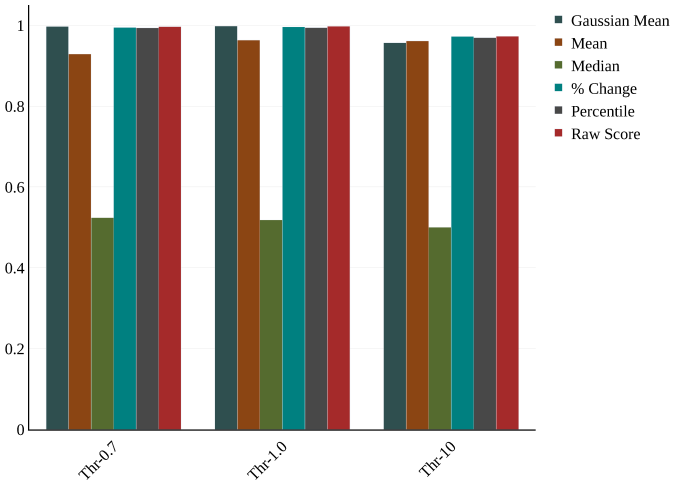

Fig. 2: *ABATe* SWaT data AUC Plots

Open/Closed) while some others had continuous values (eg. Water tank level). On preprocessing, we normalized the values from all the components to a range of 0:0 to 1:0. To study the effect of state similarity threshold hyper-parameter, we generated models using 3 different values for this parameter; 10%, 1%, and 0.7%. This threshold determines when should two states to be folded into a single state. I.e. if the state similarity threshold is 0.7%, two states are folded to a single state when the Euclidean distance between the two points is less than 0.7% of the maximum value. An event model and a context model are generated using each of these values. The learning rate hyper-parameter in the context generation phase as 0:0001 and context vector length is fixed at 100.

After the offline-learning phase, we generated 3 sets of event models and context models. The well-annotated attack dataset (which is not used during the offline phase) comprising 36 different attacks performed over 4 days is then used to evaluate *ABATe*. In the online phase, the sequences of *ABATe* states and $ABATe_{scores}$ were generated as described in section 5. Different techniques applied to the generated sequence of $ABATe_{scores}$ to detect the presence of different attacks include *gaussian-average*, *window-median*, *percentile*, and *percentage change*.

6.1.2 Results

We now present our results and compare them with other approaches published in the literature. The most common metric used to measure the performance of an anomaly detection system is the ROC (Receiver Operating Characteristic) curve that plots the sensitivity (True positive rate) on y-axis against specificity (False positive rate) on x-axis. By true positives, we mean attacks getting detected as attacks and true negatives are normal points detected as normal. Each point in the graph will be the FPR against TPR for a specific threshold. A similar measure is the AUC (Area Under the Curve) of the ROC curve. The value of AUROC will always be below 1.0 and a higher value means better TPR's at lower FPR's.

We used 2 different ways to calculate true positives and false negatives that are used to calculate precision, recall,

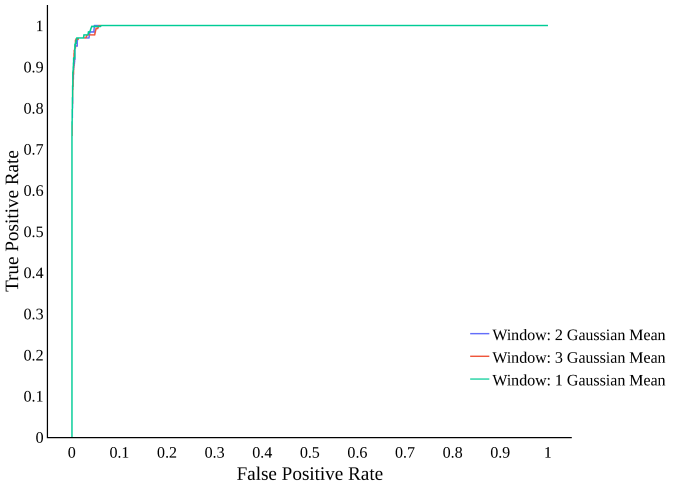

Fig. 3: *ABATe* SWaT data ROC Plots: Window Comparison using Gaussian Mean Window

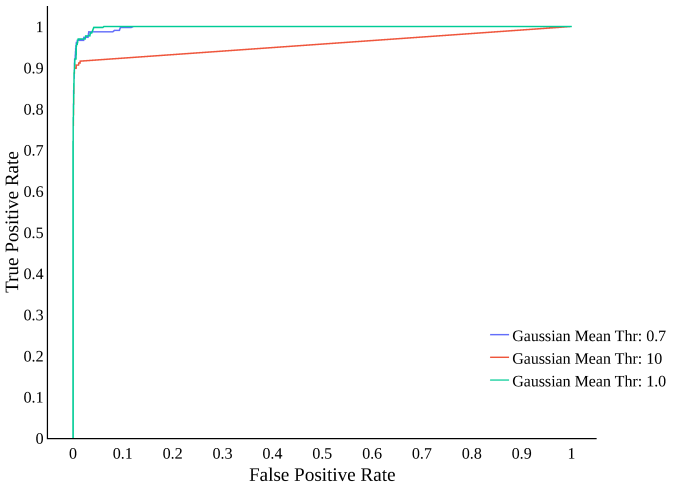

Fig. 4: *ABATe* SWaT data ROC Plots: State Transition Threshold Comparison at Window size = 1

and f1 score values. In the first method (attack-window based), we detect the attack as a whole and increment true positives or false negatives by the length of the current attack window. That is, if the current attack has a sequence of n vectors, and if the anomaly score goes above the threshold for any of these vectors, we increment true positives by n. Otherwise, we increment false negatives by n. For *ABATe*, this is a better measurement because when an attack happens the system may shift from state s_i to another valid but contextually dissimilar state s_j. *ABATe* will detect this as an anomaly. However, some further states after s_j may be contextually similar to s_j and may be reported as false negatives although the attack is already detected. In the alternative method (log-entry based), we find an anomaly score for every vector in the dataset and classify it as a true positive, true negative, false positive, or false negative according to the labels from the dataset. We report the comparison measures for both these methods and compare it against other techniques in the literature. ROC curves reported used attack-window based technique for graph generation.

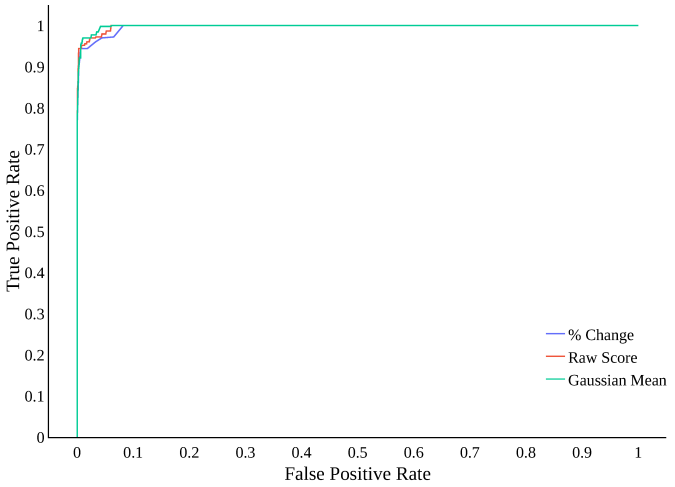

Fig. 5: *ABATe* SWaT data ROC Plots: Promising Technique ROC Curve Comparison

First, we study the effects of moving-window based approaches on the stream of $ABATe_{score}$'s generated to identify anomalies. We experimented with different moving window measures of mean, median, percentile, and gaussian mean along with raw $ABATe_{score}$'s. One other measurement with which we experimented is using the percent change of $ABATe_{score}$ for anomaly detection (Zero values for the score results in infinity values for percentage change. Hence percent change on the additive inverse of $ABATe_{score}$ values is used for detecting anomalies). Figure 2 depicts the clustered bar graph of AUROC values for these different metrics. On the x-axis of this graph, each cluster corresponds to a specific state similarity threshold and each bar in the cluster corresponds to the technique applied to aggregate $ABATe_{score}$ values. The window size used for the window-based techniques were fixed at 60.

Figure 2 shows that the best performing metric over $ABATe_{score}$ is the gaussian average. While median based techniques performed poorly, techniques like percentile, and percent change have good performances. To examine the differences more closely, we plotted the ROC curves for the most promising techniques, namely gaussian mean, percentile, and percent change in Figure 5. From this plot, we see that the gaussian mean technique performs slightly better than others where 32 attacks were detected at around 1% false positive rate.

Next, we study the effect of state transition threshold on the attack detection performance of *ABATe* using this dataset. In general, the AUC clustered plot in Figure 2 shows that maximum performance is achieved at 1% state transition threshold. Figure 4 plots the ROC curves on the SWaT attack dataset, using gaussian mean technique with different state transition thresholds (1.0%, 10.0%, and 0.7%). Lower state transition thresholds imply more number of unique states in the system but it is not translating to better results. For example, 32 attacks were detected at around 1% false positive rate for 1% state transition threshold while only 31 attacks were detected at the same false positive rate with 0.7% state transition threshold. There are two major reasons for this behavior. Firstly, when state transition thresholds

come down, many actually similar states will get split into separate states. Secondly, some of the newly created states will occur very scarcely in the dataset. As a result, *ABATe* learns only very minimum information about them from the dataset. A very high value for state transition threshold is expected to perform poorly because it compresses many semantically different states as a single state. This evaluation shows that choosing a proper state transition threshold hyper-parameter can indeed result in better performance of *ABATe*. Finally, Figure 3 plots the performance of *ABATe* with context windows 1, 2, and 3. In this case, we can see that all the three showed similar performance.

Several machine learning techniques like CNN [28], DNN [23], and GAN [34] have been used in recently published papers to detect attacks from the SWaT dataset. As we detail next, in comparison to these techniques, *ABATe* achieved better results.

Technique	Precision	Recall	F1 Score
CNN [28] (8 layers)	N/A	N/A	0.861
DNN [23]	0.98	0.68	0.80
One Class SVM [23]	0.92	0.69	0.79
PCA [34]	0.2492	0.2163	0.23
KNN [34]	0.0783	0.0783	0.08
FB [34]	0.1017	0.1017	0.1
AE [34]	0.7263	0.5263	0.61
TABOR [35]	0.86	0.788	0.82
MADGAN [34]	0.9897	0.6374	0.77
$ABATe_{gaussian}$ (log-entry based)	0.95	0.63	0.76
$ABATe_{gaussian}$ (attack-window based)	0.95	0.95	0.95

TABLE 1: Comparison of Techniques against SWaT dataset

Precision, recall, F1 score values from different techniques are reported in table 1. It can be seen that *ABATe* has one of the top reported F1 scores. The attack-window based measurements for *ABATe* has an F1 score of 0.95 with high values for both precision and recall. However, a note of caution. As Inoue et al. [23] suggest, the reported numbers for different techniques cannot always be directly compared. For instance, SVM based techniques use a window approach unlike DNN's. This argument is applicable while comparing other techniques also. Moreover, F1 score alone should not be used to compare performances because at the reported F1 scores, many techniques have low values for Recall which implies that fewer attacks are correctly identified. *ABATe* shows high numbers for precision and recall! Apart from achieving higher recall numbers, a factor that distinguishes *ABATe* from other techniques that use neural networks is its simple network used for training. This means lower training time compared to other DeepNN based techniques described in literature. Moreover, once trained, we do not need to run neural networks during online detection phase. Instead, we can store the context vectors as efficient lookup tables. This will remove the requirement for specialized circuitry in the CPS environment.

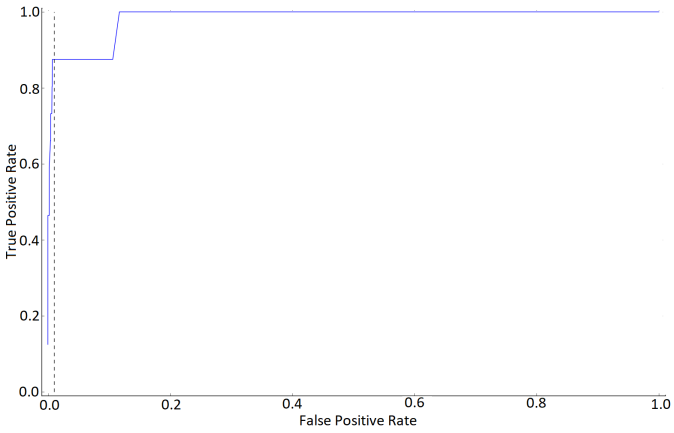

Fig. 6: *ABATe* Car data ROC curve

6.2 Automotive dataset Evaluation

Evaluations with SWaT dataset demonstrated the effectiveness of *ABATe* in detecting real attacks on cyber-physical systems. The ability of *ABATe* to detect attacks is because of its capability to abstract the CPS context from various component measurements. We demonstrate this capability using a new automotive dataset featuring 25 hours of real driving data. Besides it also demonstrates the multi-domain adaptability of *ABATe*.

A modern car with advanced systems like lane-departure warning system, adaptive cruise controls, and Anti-lock Braking System is a typical cyber-physical system with sensors, control systems, and actuators. Recent research enlists a growing number of attacks against cars. They are partly attributed to the fact that cars are becoming intelligent and many driving decisions are now transferred from drivers to control systems. Research focusing on autonomous driving will only boost these attack numbers. As described in previous research [42], a core issue with the automotive domain is their requirement to support legacy systems (owing to economic and practical limitations). For example, in some cars, hi-tech features are built on top of comparatively older CAN bus technology which lacks even basic authentication and authorization techniques. Researchers utilized this vulnerability to demonstrate many attacks.

6.2.1 Data Collection

Since existing dataset from vehicles lack enough sensor data or are focused for image processing tasks, we collected a rich dataset with data from 13 sensors, as described in table 2 from a car.

1. accelerator_pedal_position	*8. brake_pedal_status*
2. door_status	*9. engine_speed*
3. headlamp_status	*10. high_beam_status*
4. ignition_status	*11. parking_brake_status*
5. steering_wheel_angle	*12. torque_at_transmission*
6. transmission_gear_position	*13. vehicle_speed*
7. windshield_wiper_status	

TABLE 2: Components in the Automotive Dataset

Several techniques are available to collect data from cars. It includes using chipsets like ELM 327, STN1100, Arduino with CAN-BUS shield, etc. which captures raw CAN bus messages [42] and tools like OpenXC from Ford, Octane CAN bus sniffer [7], Komodo CAN bus sniffer[2], Vehicle Spy[3], SavvyCAN[4], O2OO Data logger[5], etc. for their collection and analysis. Each tool has their own advantages. For example, Vehicle Spy helps us to view all the messages in different CAN buses simultaneously. It also allows capturing and replaying different captured sessions on to the CAN bus. But it still shows raw CAN messages with raw CAN bus ID's which doesn't give any clear intuition on the semantics of different messages.

We used the OpenXC platform[6] for data collection from cars. It allows developers to unlock the rich vehicular data that can be easily exported to standard formats like JSON. OpenXC supports different hardware, that are to be connected to the OBD-II port of the vehicle, to tap data flowing through CAN bus. Some of the supported hardware include Ford Reference VI, CrossChasm C5 devices, Cross Chasm C5 BT, OpenChasm C5 cellular, CrossChasm C5 BLE, DIY chipKIT-based VI etc. In our setup, we chose to use Ford Reference VI as hardware and Android API as software for OpenXC. After choosing the vehicle specific firmware for Reference VI, a nexus 7 tablet is used to aggregate the JSON data. The collected data is a stream of discrete JSON messages. Each JSON will carry information from a specific sensor available at that time. Figure 7 shows the structure of the actual data from the Nexus 7 device. After necessary cleaning, we generated raw vectors that represent the individual states of all sensors at that time instant.


```
{"value":67.0,"name":"torque_at_transmission","a":{"a":{"name":"torque_at_
transmission"},"b":-1735469873},"timestamp":1477698358.014}
{"value":0.0,"name":"vehicle_speed","a":{"a":{"name":"vehicle_speed"},"b":
1497406494},"timestamp":1477698358.021}
{"value":0.0,"name":"accelerator_pedal_position","a":{"a":{"name":"acceler
ator_pedal_position"},"b":-951985080},"timestamp":1477698358.021}
{"value":0.0,"name":"engine_speed","a":{"a":{"name":"engine_speed"},"b":11
91599776},"timestamp":1477698358.027}
{"value":0.0,"name":"steering_wheel_angle","a":{"a":{"name":"steering_whee
l_angle"},"b":-1273841975},"timestamp":1477698358.099}
{"value":"off","name":"ignition_status","a":{"a":{"name":"ignition_status"
},"b":1643942958},"timestamp":1477698358.103}
{"value":false,"name":"parking_brake_status","a":{"a":{"name":"parking_bra
ke_status"},"b":949311045},"timestamp":1477698358.103}
```


Fig. 7: Automotive data from OpenXC VI

We collected about 25 hours of real driving data with a total of close to 1000 miles of driving. We made sure that the dataset contains data from different driving conditions which include city drives, highway drives, short drives, hill road drives, and short shopping drives as described in table 3.

6.2.2 Test Setup

The main objective of using this automotive dataset with *ABATe* is to study our technique's capability to abstract context from real-world cyber-physical system components. Also, the study demonstrates its ability to adapt to varied cyber-physical system settings. To evaluate *ABATe*, we preprocessed the real driving data and converted the collected JSON to raw vectors, $r\forall_t$. We used the state similarity threshold as 5% and we were able to generate 365 unique states. We used the *ABATe* implementation pipeline with the learning rate as 0.0001, context vector length as 100, and used Adams optimizer for training the neural network to generate the context model.

Drive Condition	Miles Driven	Time Duration(hrs)
Hill Drive	268.51	5.51
Hiway Drive	609.06	14.24
Short/City Drive	109.11	5.67
Total	986.68	25.42

TABLE 3: Automotive Dataset Characterization

6.2.3 Context Abstraction Evaluation

The ability of *ABATe* to detect real-world attacks is demonstrated using the SWaT dataset. In order to understand how the generated $ABATe_{score}$ can identify deviations from perceived normal using observed states, it is critical to understand what information is captured in the context vectors and their semantics. In our evaluations with the automotive dataset, we study, specifically, the capability of *ABATe* to abstract contextual information into context vectors. We first generated the event model and context model using the hyper-parameters described in the test setup. On putting the generated model against the existing training dataset, the false positive was below 2%.

To test *ABATe*'s ability to abstract context, we mapped the semantics of event sequences to context vectors in our experiments with the automotive dataset. In our first experiment, we chose 2 sets of logically dissimilar states. The first set, S_{low_speed} consisted of all those states which have speed in the range of 0 to 10 miles per hour and the second set, S_{high_speed} comprises the set of all states with speeds above 60 miles per hour. Various laws of physics constraints the speed behavior and will make sure any two states ($s_i \in S_{low\_speed}$ and $s_j \in S_{high\_speed}$), cannot occur in the same context. We evaluate if *ABATe* can indeed capture this knowledge in the context vectors using the training data. Since we manually handpicked these states, we expect the similarity of the context vectors, $ContextSim$ to be very low.

On running this experiment using the current setup, we found 78 instances in the set S_{low_speed} and 134 instances in set S_{high_speed}. Out of the 10452 combinations in the set $S_{low\_speed} \times S_{high\_speed}$, 98.9% of combinations had a $ContextSim$ score of larger than 0.8 where 2.0 is the maximum possible score and none of the combinations had a $ContextSim$ score of less than 0.6. This shows that *ABATe* learned that the states from S_{low_speed} and S_{high_speed} are indeed contextually dissimilar.

Next experiment is to find the compliment of this test, that is picking up 2 states from the same set ($S_{high\_speed} \times S_{high\_speed}$) and finding their anomaly score. However, only 1.6% had a $ContextSim$ score less than 0.1. This implies that only 1.6% of the instances are highly similar. The state pair (s_i, s_j) with maximum dissimilar states (pair with max-

2. https://www.totalphase.com/products/komodo-canduo/
3. http://store.intrepidcs.com/Vehicle-Spy-p/vspy-3-ent.htm
4. http://www.savvycan.com/
5. https://www.vanheusden.com/O2OO/
6. http://openxcplatform.com/

imum *ContextSim* score) is described in table 4. It indeed shows that *ABATe* can discern the context because even with a high similarity in the *vehicle_speed*, the two states were deemed dissimilar because the *steering_wheel_angle* of these two states are pointing to different directions (negative value implies steering turned towards left and positive value implies steering turned towards right) and the *transmission_gear_position* is also different.

Component	s_i	s_j
vehicle_speed	72.839996	70.959999
door_status	0.0	0.0
accelerator_pedal_position	35.200001	0.0
ignition_status	2.0	2.0
torque_at_transmission	251.0	75.0
steering_wheel_angle	0.400024	-1.599976
parking_brake_status	0.0	0.0
high_beam_status	0.0	0.0
brake_pedal_status	0.0	1.0
headlamp_status	1.0	1.0
windshield_wiper_status	1.0	0.0
engine_speed	2738.0	1386.0
transmission_gear_position	4.0	6.0

TABLE 4: Comparing s_i *2* S_{high_speed} and s_j *2* S_{high_speed} with large $ABATe_{score}$

We performed one more experiment to testify the above capability. In this experiment, we aggregated two sets $S_{gear=1}$ which is the set of all states where *transmission_gear_position* is 1 and $S_{gear=6}$ which is the set of all states with *transmission_gear_position* is 6. Even though a change from gear 1 to gear 6 technically possible, while driving a car it is against the domain etiquette. In this experiment, $S_{gear=1}$ had 129 states and $S_{gear=6}$ had 25 states. Among the unique 3225 state pairs, 99% have a *ContextSim* greater than 0.8 and none of the state pairs scored less than 0.7.

It should be noted that we use hot vectors before training the context model which assume no relationships between any 2 states. Thus, these experiments prove that *ABATe* captures even complex contextual information involving multiple sensors into context vectors using perceived normal data.

6.2.4 Simulated Attack Detection

In the SWaT dataset discussed earlier, real attacks were conducted against the experimental setup of the sewage plant. However, in this dataset, real attacks cannot be mounted – the car cannot be driven in a way that would simulate an attack. Considering the legality and danger of exposing a real car to different attack scenarios, we simulated different attack scenarios by injecting false data into the real data collected from cars. In total, we simulated 11 different attacks. Broadly, we simulated two classes of attacks. In the first type, data from only one sensor is manipulated creating sudden spikes in speed, RPM, etc. It created states which can never occur during the normal operation of a car (single point anomalies). The next type of attacks is more complex in which existing states, but contextually dissimilar existing states are randomly injected into the real data stream. In this class of attacks, the injected states are valid states but are out of context. Figure 6 represents the corresponding ROC curve and yielded very good results by detecting most of the attacks with lesser false positive rates.

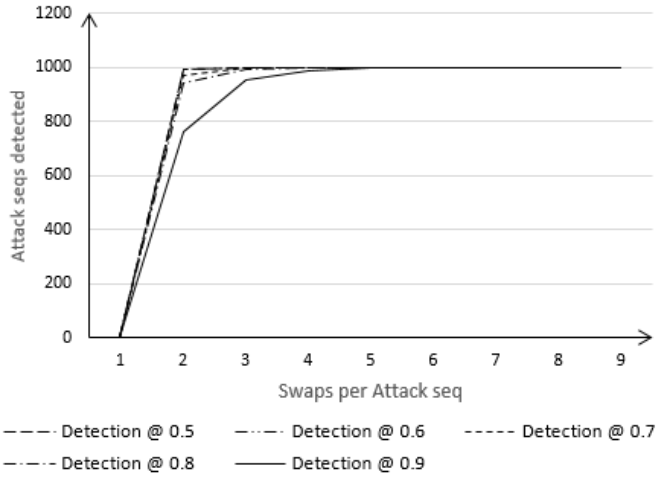

Fig. 8: *ABATe* Injection Detection Performance

Another broad category of attacks common in cyber-physical systems is FDIA (False Data Injection Attacks) in which false data is injected into the system randomly. Some of the FDIAs include the use of automated tools to inject packets into the system rather than handcrafting them. To simulate this scenario, we first extracted a normal sequence from the real-world car data. Then, we used normal distribution to choose 2 states from the subsequence randomly and swapped them. The swapping is done k times to generate a single attack sequence. We generated 1000 such attack sequences and tested it against the perceived normal *ABATe* model. The number of attack sequences detected is plotted against the number of swaps in Figure 8 when the $_{anomaly}$ is varied. Each line represents a specific value for $_{anomaly}$. It is observed that when the number of swaps increase, almost all the attacks are detected which demonstrate *ABATe*'s ability to detect such attacks.

6.3 Synthetic Dataset Evaluation

Sections 6.1 and 6.2 evaluated the performance of *ABATe* against two real-world datasets. In this section, we use a synthetically generated sequence of states with well-defined probabilities to study the intricate effects of context window used in the offline-learning phase of *ABATe*.

6.3.1 Synthetic Dataset Generation

HMM's (Hidden Markov Models) have been used to generate state sequences with fixed probabilities [12]. For the generation of a known sequence, we developed an HMM with 36 states and 36 different observations corresponding to each of these states. Each state in our state diagram will emit a fixed observation, its state number, with maximum probability. The state diagram for the HMM model we created is described in figure 10. There are 6 hexagons named A to F and each corner corresponds to 36 different states from 0 through 35. Each edge in the state diagram corresponds to a non-zero bi-directional state transition probability. We use

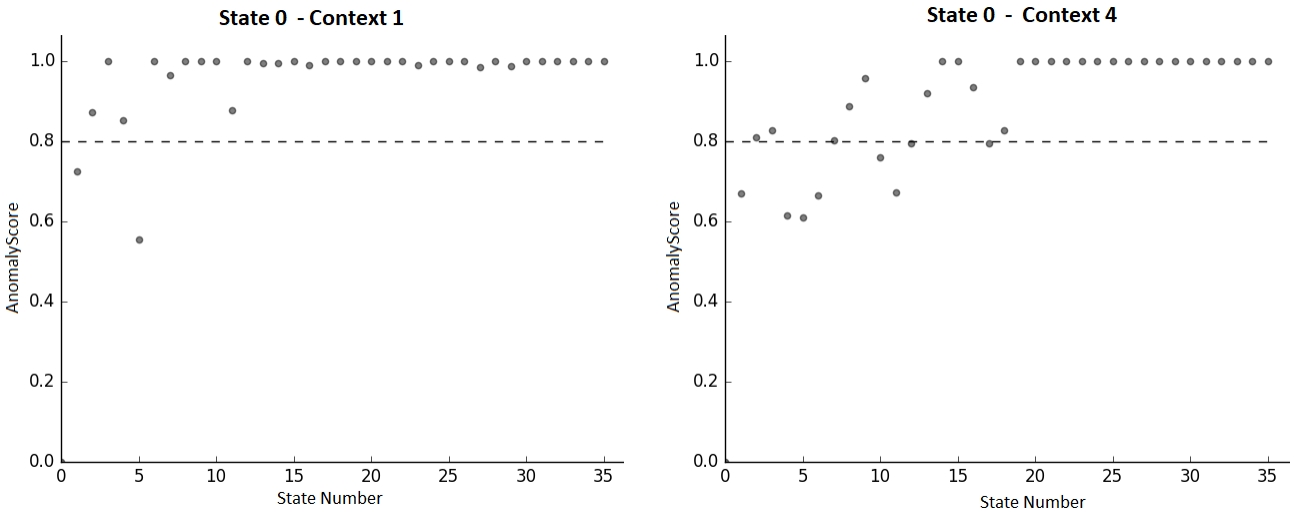

Fig. 9: Comparison of State 0 Anomaly Scores with Context 1 and Context 4

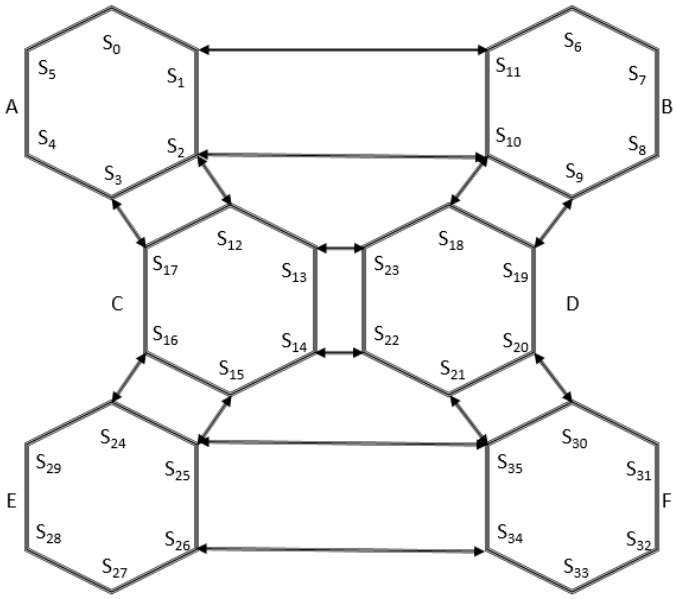

Fig. 10: Synthetic Data Generation: State Diagram

three different probabilities for this network. $_{ss}$ representing the probability of each state to itself, $_{sa}$ representing transition probability of moving to another state in the same hexagon and $_{ia}$ representing transition probabilities for moving from one hexagon to another if there is an edge connecting the corner to another hexagon's corner. For example, S_0 ! $S_1; S_6$! $S_7; S_{30}$! S_{31} etc. have the same probability $_{sa}$, and S_1 ! $S_{11}; S_{14}$! S_{22} etc. have the transition probability $_{ia}$. By tweaking these three variables we generate different sequences. If there is no edge connecting two vertices, they have zero probabilities. For instance S_1 ! $S_3; S_4$! S_{29}; etc. have zero chance in the sequence. We used Matlab's HMM tool kit to generate the sequence from the state transition probability matrix and emission probability matrix.

For evaluations, we generated 3 sets of sequences; sequence with transition probabilities $_{ia}$ = $_{sa}$ = $_{ss}$, sequence with $_{ia}$ greater than $_{sa}$ and $_{ss}$, and sequence with $_{sa}$ greater than $_{sa}$ and $_{ss}$. We trained each of these sequences of states using *ABATe* with window sizes 1 & 4, and generated 6 different models. To generate anomalous sequence, we added more edges to the existing state transition diagram and regenerated the sequence. For example, adding an edge from s0 ! s6 and regenerating the sequence will produce a sequence with s0 ! s6 in addition to the previous state transitions. We trained *ABATe* using the synthetic sequence and used the anomalous sequences to evaluate its performance. We were able to detect anomalous state transitions injected with 100% accuracy (since the number of states is small and their transitions are constrained).

6.3.2 Context Window Evaluation

In this section, we study the behavior of context windows in *ABATe*. As we are using only existing states, the point anomaly scores are irrelevant in Eqn: 7 and only *ContextSim* value need to be considered. To understand the semantics learned by *ABATe* when the context window is modified, we first calculated the *ContextSim* score of each state s_i against all the states in S for models generated with context window as 1 and 4. When the context window is 1 only the states immediately connected using the edges should have *ContextSim* score below a specific threshold. However, when we increase the windows size more states should come into context.

End State	Path
S_1	S_0 ! S_1
S_4	S_0 ! S_5 ! S_4
S_5	S_0 ! S_5
S_6	S_0 ! S_1 ! S_{11} ! S_6
S_{10}	S_0 ! S_1 ! S_2 ! S_{10}
S_{11}	S_0 ! S_1 ! S_{11}

TABLE 5: Synthetic Dataset State transition path for State S_0

Figure 9 presents the *ContextSim* scores for the state S_0 with context windows sizes 0 and 4. From the state diagram in Figure 10, we can see that S_0 is only directly connected

to s_1 and s_5. As a result, we can see the *ContextSim* scores for s_0, s_5, and s_1 are below the threshold values. However, when the context window size is 4, many other states became non-anomalous. This is because many other states which are 2, 3, and 4 edges from the current state are also added into the context as a result of increasing the context window size. For example, the states which came into context clearly are s_1, s_4, s_5, s_6, and s_{10}. The states s_{11}, s_7, s_{12} , and s_{17} also just made the list. The list of edges which put these states into the context of s_0 is presented in table 5. However, some other states like s_9 did not come into context even though it is 4 edges away. We believe that this should be because of the fact that the intermediate edges s_2 and s_{10} in the path from s_0 ! s_9 have more branching options and hence when the normal sequence was generated the number of instances which had that sequence is much lower compared to others. From these experiments, we can understand how the semantics of *ABATe*'s training is altered with change in context window size.

This will be an important feature to add robustness to our technique in the real world. There would be many cyber-physical systems where some states would be missing from the streaming input data. Such occurrences can happen because of inherent noise or difference in sampling intervals during training and monitoring. It is also shown in Figure 3 from SWaT dataset evaluation that the false positive rates are comparable when the window sizes are modified. Hence, slightly adjusting the window size combat such missing states without much increasing the FPR's.

6.4 Time Complexity

In this section, we discuss the time complexity of using *ABATe* in real-world systems. Since the offline learning phase needs to be run only less often, we focus on the online monitoring phase here. As discussed in section 4, the first task is the generation of raw vector, rv_i^0 which is a straight forward function with complexity of $O(1)$. The next task is to choose a $\hat{rv}_i^0$ corresponding to it. Time complexity for calculating EU_i and choosing one state from it is $O(|S|)$, ($|S|$ is the number of states used in the model) since the number of states is a constant during online monitoring. Hence the overall complexity of one decision making is $O(|S|)$ and makes *ABATe* practical for real-world applications.

7 Conclusion & Future work

In this paper, we propose *ABATe*, a technique to detect anomalies in smart cyber-physical systems. It abstracts the normal working behavior of a CPS using the abundantly available operational data into context vectors, and uses them to detect various point and context anomalies in that domain. Some of *ABATe*'s features include multi-domain adaptability, domain context abstraction, and accommodation of occasional bad data in the training operational data. *ABATe* detected around 32 out of 36 attacks with a false positive rate of 1% with SWaT dataset, which consists of data from 51 sensors from a scaled down sewage water treatment plant. Apart from demonstrating *ABATe*'s ability to detect attacks in real-world CPS, we aggregated a dataset from cars and showed *ABATe*'s ability to abstract context by comparing the anomaly scores of logically indifferent states. However, the false positive rate of 1% is not considered low enough in some specific cases where the frequency of states is very high. In the future, several other research in the field of anomaly detection could be coupled with *ABATe* to bring down this false positive rate. One promising addition to *ABATe* would be to use other popular techniques based on k-nearest neighbor, local out-lier factor, etc. for point anomaly detection. Learning state boundaries is another interesting avenue to explore.

Acknowledgments

We thank IBM for their gift that partly supported this research.

References

[1] S. Adepu and A. Mathur. Distributed attack detection in a water treatment plant: Method and case study. *IEEE Transactions on Dependable and Secure Computing*, 2018. doi: 10.1109/TDSC.2018.2875008.

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 2017.

[3] Victor Berger. Anomaly detection in user behavior of websites using Hierarchical Temporal Memories: Using Machine Learning to detect unusual behavior from users of a web service to quickly detect possible security hazards. Master's thesis, KTH, School of Computer Science and Communication (CSC)., Stockholm Sweden, 2017.

[4] Anatolij Bezemskij, George Loukas, Richard J Anthony, and Diane Gan. Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle. In *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, pages 61–68. IEEE, 2016.

[5] Anatolij Bezemskij, George Loukas, Diane Gan, and Richard J Anthony. Detecting cyber-physical threats in an autonomous robotic vehicle using bayesian networks. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 98–103. IEEE, 2017.

[6] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[7] P Borazjani, C Everett, and Damon McCoy. Octane: An extensible open source car security testbed. In *Proceedings of the Embedded Security in Cars Conference*, page 60, 2014.

[8] Suratna Budalakoti, Ashok N Srivastava, Ram Akella, and Eugene Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. Technical report, NASA TM-2006-214553, NASA Ames Research Center, 2006.

[9] Suratna Budalakoti, Ashok N Srivastava, and Matthew E Otey. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):101–113, 2009.

[10] Soumen Chakrabarti, Sunita Sarawagi, and Byron Dom. Mining surprising patterns using temporal description length. In *VLDB*, volume 98, pages 606–617, 1998.

[11] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[12] Varun Chandola, Varun Mithal, and Vipin Kumar. Comparative evaluation of anomaly detection techniques for sequence data. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 743–748. IEEE, 2008.

[13] Javier Alvarez Cid-Fuentes, Claudia Szabo, and Katrina Falkner. Adaptive performance anomaly detection in distributed systems using online svms. *IEEE Transactions on Dependable and Secure Computing*, 2018.

[14] Gyorgy Dan and Henrik Sandberg. Stealth attacks and protection schemes for state estimators in power systems. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 214–219. IEEE, 2010.

[15] Robert J Daverman and Gerard Venema. *Embeddings in manifolds*, volume 106. American Mathematical Soc., 2009.

[16] German Florez-Larrahondo, Susan M Bridges, and Rayford Vaughn. Efficient modeling of discrete events for anomaly detection using hidden markov models. In *International Conference on Information Security*, pages 506–514. Springer, 2005.

[17] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *International Conference on Critical Information Infrastructures Security*, pages 88–99. Springer, 2016.

[18] Jonathan Goh, Sridhar Adepu, Marcus Tan, and Zi Shan Lee. Anomaly detection in cyber physical systems using recurrent neural networks. In *High Assurance Systems Engineering (HASE), 2017 IEEE 18th International Symposium on*, pages 140–145. IEEE, 2017.

[19] Dieter Gollmann. Security for cyber-physical systems. In *International doctoral workshop on Mathematical and Engineering Methods in Computer Science*, pages 12–14. Springer, 2012.

[20] Dieter Gollmann, Pavel Gurikov, Alexander Isakov, Marina Krotofil, Jason Larsen, and Alexander Winnicki. Cyber-physical systems security: Experimental analysis of a vinyl acetate monomer plant. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, pages 1–12. ACM, 2015.

[21] Junho Hong, Chen-Ching Liu, and Manimaran Govindarasu. Integrated anomaly detection for cyber security of the substations. *IEEE Transactions on Smart Grid*, 5(4):1643–1653, 2014.

[22] Tobias Hoppe and Jana Dittman. Sniffing/replay attacks on can buses: A simulated attack on the electric window lift classified using an adapted cert taxonomy. In *Proceedings of the 2nd workshop on embedded systems security (WESS)*, pages 1–6, 2007.

[23] Jun Inoue, Yoriyuki Yamagata, Yuqi Chen, Christopher M Poskitt, and Jun Sun. Anomaly detection for a water treatment system using unsupervised machine learning. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1058–1065. IEEE, 2017.

[24] Austin Jones, Zhaodan Kong, and Calin Belta. Anomaly detection in cyber-physical systems: A formal methods approach. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 848–853. IEEE, 2014.

[25] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Data mining, fifth IEEE international conference on*, pages 8–pp. Ieee, 2005.

[26] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):20, 2019.

[27] Louis Kratz and Ko Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1446–1453. IEEE, 2009.

[28] Moshe Kravchik and Asaf Shabtai. Detecting cyber attacks in industrial control systems using convolutional neural networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 72–83. ACM, 2018.

[29] Marina Krotofil, Jason Larsen, and Dieter Gollmann. The process matters: Ensuring data veracity in cyber-physical systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 133–144. ACM, 2015.

[30] Terran Lane, Carla E Brodley, et al. Sequence matching and learning in anomaly detection for computer security. In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, pages 43–49, 1997.

[31] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1939–1947. ACM, 2015.

[32] Pavel Laskov, Konrad Rieck, Christin Schäfer, and Klaus-Robert Müller. Visualization of anomaly detection using prediction sensitivity. In Hannes Federrath, editor, *Sicherheit 2005: Sicherheit - Schutz und Zuverlässigkeit, Beiträge der 2. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.v. (GI), 5.-8. April 2005 in Regensburg*, volume P-62 of *LNI*, pages 197–208. GI, 2005.

[33] Beibei Li, Rongxing Lu, Wei Wang, and Kim-Kwang Raymond Choo. Distributed host-based collaborative detection for false data injection attacks in smart grid cyber-physical system. *Journal of Parallel and Distributed Computing*, 103:32–41, 2017.

[34] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, pages 703–716. Springer, 2019.

[35] Qin Lin, Sridha Adepu, Sicco Verwer, and Aditya Mathur. Tabor: a graphical model-based approach for anomaly detection in industrial control systems. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 525–536. ACM, 2018.

[36] Chao Liu, Sambuddha Ghosal, Zhanhong Jiang, and Soumik Sarkar. An unsupervised spatiotemporal graphical modeling approach to anomaly detection in distributed cps. In *Cyber-Physical Systems (ICCPS), 2016 ACM/IEEE 7th International Conference on*, pages 1–10. IEEE, 2016.

[37] Guoying Liu, Timothy K McDaniel, Stanley Falkow, and Samuel Karlin. Sequence anomalies in the cag7 gene of the helicobacter pylori pathogenicity island. *Proceedings of the National Academy of Sciences*, 96(12):7011–7016, 1999.

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[39] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. Technical report, Blackhat 2015, 2015.

[40] Robert Mitchell and Ing-Ray Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)*, 46(4):55, 2014.

[41] Robert Mitchell and Ray Chen. Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems. *IEEE Transactions on Dependable and Secure Computing*, 12(1):16–30, 2015.

[42] S. N. Narayanan, S. Mittal, and A. Joshi. Obdsecurealert: An anomaly detection system for vehicles. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6, May 2016.

[43] Clifford Neuman. Challenges in security for cyber-physical systems. In *DHS workshop on future directions in cyber-physical systems security*, pages 22–24. Citeseer, 2009.

[44] Timothy J O'Shea, T. Charles Clancy, and Robert W. McGwier. Recurrent neural radio anomaly detection, 2016.

[45] Miroslav Pajic, James Weimer, Nicola Bezzo, Paulo Tabuada, Oleg Sokolsky, Insup Lee, and George J Pappas. Robustness of attack-resilient state estimators. In *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014)*, pages 163–174. IEEE Computer Society, 2014.

[46] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. *International journal of computer applications*, 175(4):7–9, 2017.

[47] Yan Qiao, XW Xin, Yang Bin, and S Ge. Anomaly intrusion detection method based on hmm. *Electronics letters*, 38(13):663–664, 2002.

[48] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, Zahra. Moayed, and Reinhard Klette. Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, 172:88 – 97, 2018.

[49] Markus Schneider, Wolfgang Ertel, and Fabio Ramos. Expected similarity estimation for large-scale batch and streaming anomaly detection. *Machine Learning*, 105(3):305–333, 2016.

[50] Siddharth Sridhar, Adam Hahn, and Manimaran Govindarasu. Cyber–physical system security for the electric power grid. *Proceedings of the IEEE*, 100(1):210–224, 2011.

[51] Chris Valasek and Charlie Miller. Adventures in automotive networks and control units. In *IOActive*, 2014.

[52] Tuan Vuong, Avgoustinos Filippoupolitis, George Loukas, and Diane Gan. Physical indicators of cyber attacks against a rescue robot. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 338–343. IEEE, 2014.

[53] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 133–145. IEEE, 1999.