Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it’s important to you. Thank you.
ABATe: Automatic Behavioral Abstraction Technique to Detect Anomalies in Smart Cyber-Physical Systems

Sandip 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 Automatic Behavioral Abstraction Technique 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” 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 (Automatic Behavior Abstraction Technique) 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...
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\(e_{score}\) 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. Sabokrouou 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. Krotolif 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 \textit{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 \textit{ABATe} can automatically learn general constrained behavior of CPS systems. It is capable of abstracting the domain’s normal behavior in the form of vectors using its operational data and makes \textit{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 \textit{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 \textbf{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 \textbf{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 \textit{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 \textit{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 \( x_i \in X \) 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 \( t \), \( r\bar{v}_t \) (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 \( r\bar{v}_t \) 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.

\[
X = \{ x_1, x_2, ..., x_i, ..., x_n \}
\]  
\[
r\bar{v}_t = (v_{1,t}, v_{2,t}, ..., v_{i,t}, ..., v_{n,t})
\]
In this task, ABATe learns context vectors that abstract the perceived normal behavior of the CPS for each state \( h_{v_i} \) in \( HS \). To learn context vectors, ABATe finds an embedding for each state \( h_{v_i} \) 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, \( h_{v_i} \in HS \) using neural networks and CPS normal data. These embeddings encapsulate the distributional behavior and encode the contextual information with them. For example, let \( c_i, c_j \) be the learned embeddings corresponding to \( h_{v_i}, h_{v_j} \). Then, the vector similarity between \( c_i \) & \( c_j \) will be high if \( h_{v_i} \) & \( h_{v_j} \) 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 \( h_{v_i} \) into \( c_i \) as presented in Eq: 4 and Eq: 5. The network has an input layer with size equal to the number of states in \( HS \) \((HS_{\text{size}} = | HS |)\). They are connected to a fully-connected neural layer with input size as \( HS_{\text{size}} \) and output size as the context vector length, \( C_{\text{size}} \) (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 (\( C_{\text{size}} \)) and output size as \( HS_{\text{size}} \). The soft-max activation function implements the softmax 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 \( c_i \). The \( c_i \) is a condensed representation of \( h_{v_i} \) 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:
\[
\begin{align*}
\vec{c_i} &= f(h_{v_i}) \\
f(x) &= Wx + B
\end{align*}
\]
where,
\[
\begin{align*}
h_{v_i} &: \text{Hot Vector corresponding to } s_i \\
W &: \text{Matrix with size } HS_{\text{size}} \times C_{\text{size}} \\
B &: \text{vector with size } C_{\text{size}}
\end{align*}
\]
Output Layer:
\[
\begin{align*}
\text{Output} &= \text{SoftmaxAct}(f'(c_i)) \\
f'(x) &= W'x + B' \\
\text{SoftmaxAct}(x) &= \frac{e^{x_i}}{\sum_{h_{v_i} \in HS} e^{x_h}}
\end{align*}
\]
where,
\[
\begin{align*}
c_i &: \text{Context Vector corresponding to } s_i \\
W &: \text{Matrix with size } C_{\text{size}} \times HS_{\text{size}} \\
B &: \text{vector with size } HS_{\text{size}}
\end{align*}
\]
To train the network, we introduce a new variable \( ch_{v_i} \) corresponding to each \( h_{v_i} \). Each \( ch_{v_i} \) is a hot vector and assumes no relationship between any existing \( h_{v_i} \) and is...
used only for training the neural network. We also introduce a context window of size $cw_{\text{size}}$ while training. It defines the influence of a state to its preceding states. For example, if $cw_{\text{size}} = 2$, 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 (input, output). We generate these tuples from the sequence of normal operational data of a cyber-physical system. Let $hv_1, hv_2, ..., hv_i, hv_{i+1}, ...$ be a sequence which appears in the normal operation of the CPS. For each $hv_i$, we generate tuples $(hv_{i-k}, hv_j), k=0$ to $cw_{\text{size}}$. For example, we generate tuples $(hv_{i-2}, hv_i), (hv_{i-1}, hv_i)$ and $(hv_i, hv_i)$ to $TrainSet$ if $cw_{\text{size}} = 2$. On training, the network will maximize the log probability of $hv_i$, given $hv_{i-2}$. The intuition is that the neural network will bring all the context vectors $hv_i$ which appear inside the context window $cw_{\text{size}}$ 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 offline-stage operations are presented in algorithm 1.

**Algorithm 1 Off-line Learning Phase**

**Input:**

$RV_{\text{seq}} = \bar{rv}_1, \bar{rv}_2, ..., \bar{rv}_n, ...$

**Output:**

StateMdl: Raw vector to State vector mapping
CtxMdl: State vector to Context vector mapping

1. **procedure** **LEARNING PHASE**(RS)

2. State Vector Generation
   1. $S \leftarrow \text{FoldSimilarVectorsToOne}(RV_{\text{seq}}, \tau_{\text{sim}})$
   2. $HS \leftarrow \text{GenerateHotVector}(S)$

3. Context Vector Generation
   1. $TrainSet \leftarrow []$
   2. For each window $\bar{rv}_1, ..., \bar{rv}_n, ...$ in $RV_{\text{seq}}$
   3. Append $(hv_{i-k}, hv_i, k=0$ to $cw_{\text{size}}$ to $TrainSet$
   4. $CtxMdl \leftarrow \text{TrainContextVectors}(HS, TrainSet)$
   5. $c_i \leftarrow \text{GetContextVector}(CtxMdl, s_i)$

**4.2 On-line Monitoring Phase**

In the off-line phase, we created context vectors $c_i$ corresponding to all state hot vectors $hv_i \in HS$ and encapsulated the normal behavior of the cyber-physical system. The online monitoring phase aggregates live inputs from different components in the system and generates an $ABAT e_{\text{score}}$ 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, $InputRV_{\text{seq}} = \bar{rv}_1', \bar{rv}_2', \bar{rv}_3', ..., \bar{rv}_n'$, using inputs from different components in the CPS, where $\bar{rv}_i'$ is the input vector at time $t$. As described in section 4.1.1, each $\bar{rv}_i'$ will be a point the same $n$-dimensional space. We choose a state $s_i' \in S$ corresponding to each $\bar{rv}_i'$ such that the Euclidean distance between $s_i'$ and $\bar{rv}_i'$ is minimum. Now a sequence of states $InputState_{\text{seq}} = s_1', s_2', ..., s_n', ...$ will be generated where $s_i'$ is the chosen state corresponding to each $\bar{rv}_i'$. Let $EU_i$ be the Euclidean distance between a raw vector $\bar{rv}_i'$ and $\bar{rv}_i$ corresponding to $s_i'$. In all normal cases, $EU_i$ 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 $EU_i$ 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 $InputCV_{\text{seq}} = c_1', c_2', ..., c_n', ...$ be the context vector sequence corresponding to $InputState_{\text{seq}}$, where $c_i'$ is the context vector corresponding to state $s_i'$. In ABATe, we use the cosine distance (Eqn: 6) of the current context vector $c_i'$ and the previous context vector $c_i-1'$ to detect context anomalies and is denoted by $ContextSim_{\text{ct}}$. The $ContextSim_i$ determines the appropriateness of a vector to its context or nearby vectors. A lower score implies that the current vector is very inappropriate 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 $EU_i$ and $ContextSim_i$ are related. When the value of $EU_i$ is high, the confidence in that state is very low and the corresponding $ContextSim_i$ score also should be considered with lower confidence. Hence to generate a combined score to detect point anomalies and contextual anomalies, we use a combined score as described in Eqn: 7. The first component in the $ABAT e_{\text{score}}$ corresponds to the dissimilarity of current raw vector $\bar{rv}_i'$ to the known states. Since we used 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 $EU_i$ increases the likelihood of point and contextual anomalies. The second component indicates the contextual appropriateness of the new state. A low value for $ABAT e_{\text{score}}$ 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 $ABAT e_{\text{score}}$ for discerning anomalies by choosing an empirically estimated detection threshold $\tau_{\text{anomaly}}$. The operations of this stage are described in algorithm 2.

\[
ContextSim_i = 1 - \frac{\|c_i' - c_i-1'\|}{\|c_i'\| \cdot \|c_i-1'\|}
\]

where,

$c_i', c_i-1'$: Context Vectors corresponding to $s_i'$ and $s_i-1'$

(6)
Our implementation of Algorithm 2 line phases as enumerated below.

### 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.

### 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, $\tau_{sim}$ (a hyper-parameter). Once similar states are aggregated, they are labeled randomly.

### 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.

### 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 $\vec{r}^*_t$. Then a state label $\vec{s}_t$ is determined from the event model by choosing the state having the closest Euclidean distance from $\vec{r}^*_t$. The vector $\vec{c}_t$ corresponding to $\vec{s}_t$ 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.

### 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 $\tau_{anomaly}$. 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 $\tau_{anomaly}$ needs to be empirically estimated considering how critical the CPS setup is. If the CPS is critical, a conservative low value of $\tau_{anomaly}$ should be chosen while non-critical systems can afford higher thresholds. We found that the average based techniques yielded best performance.

### 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 ABAT\textsubscript{c}\textsubscript{scores}’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 database (off-line Learning phase). The values from different sensors had different ranges. For example, some of them are binary (eg. valve status; 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 ABAT\textsubscript{c}\textsubscript{scores} were generated as described in section 5. Different techniques applied to the generated sequence of ABAT\textsubscript{c}\textsubscript{scores} to detect the presence of different attacks include gaussian-average, window-median, percentile, and percentage change.

6.1.2 Results

Fig. 2: ABATe SWaT data AUC Plots

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

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.

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

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.

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.

<table>
<thead>
<tr>
<th>Sensor Category</th>
<th>Sensor Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerator pedal</td>
<td>Pedal Position</td>
</tr>
<tr>
<td>Door</td>
<td>Status</td>
</tr>
<tr>
<td>Headlamp</td>
<td>Status</td>
</tr>
<tr>
<td>Ignition</td>
<td>Status</td>
</tr>
<tr>
<td>Steering wheel</td>
<td>Angle</td>
</tr>
<tr>
<td>Transmission</td>
<td>Gear position</td>
</tr>
<tr>
<td>Windshield</td>
<td>Wiper status</td>
</tr>
<tr>
<td>Engine</td>
<td>Speed</td>
</tr>
<tr>
<td>High beam</td>
<td>Status</td>
</tr>
<tr>
<td>Parking brake</td>
<td>Status</td>
</tr>
<tr>
<td>Torque</td>
<td>At transmission</td>
</tr>
<tr>
<td>Vehicle</td>
<td>Speed</td>
</tr>
</tbody>
</table>

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 its 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.

![Fig. 7: Automotive data from OpenXC VI](image)

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 pre-processed the real driving data and converted the collected JSON to raw vectors, \( \vec{r}_\text{v} \). 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.

### 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 \( \text{ABAT}_\text{e-score} \) 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, \( S_{\text{low-speed}} \) consisted of all those states which have speed in the range of 0 to 10 miles per hour and the second set, \( S_{\text{high-speed}} \) 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 \((s_i \in S_{\text{low-speed}} \text{ and } s_j \in S_{\text{high-speed}})\), 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, \( \text{ContextSim} \) to be very low.

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

Next experiment is to find the compliment of this test, that is picking up 2 states from the same set \( (s_i \in S_{\text{high-speed}} \times S_{\text{high-speed}}) \) and finding their anomaly score. However, only 1.6% had a \( \text{ContextSim} \) 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-

### TABLE 3: Automotive Dataset Characterization

<table>
<thead>
<tr>
<th>Drive Condition</th>
<th>Miles Driven</th>
<th>Time Duration (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hill Drive</td>
<td>268.51</td>
<td>5.51</td>
</tr>
<tr>
<td>Hiway Drive</td>
<td>609.06</td>
<td>14.24</td>
</tr>
<tr>
<td>Short/City Drive</td>
<td>109.11</td>
<td>5.67</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>986.68</strong></td>
<td><strong>25.42</strong></td>
</tr>
</tbody>
</table>

---

3. [http://store.intrepidics.com/Vehicle-Spy-p/vspy-3-ent.htm](http://store.intrepidics.com/Vehicle-Spy-p/vspy-3-ent.htm)
5. [https://www.vanheusden.com/O2OO/](https://www.vanheusden.com/O2OO/)
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.

<table>
<thead>
<tr>
<th>Component</th>
<th>(s_i)</th>
<th>(s_j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>vehicle_speed</td>
<td>72.839996</td>
<td>70.999999</td>
</tr>
<tr>
<td>door_status</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>accelerator_pedal_position</td>
<td>35.200001</td>
<td>0.0</td>
</tr>
<tr>
<td>ignition_status</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>torque_at_transmission</td>
<td>251.0</td>
<td>75.0</td>
</tr>
<tr>
<td>steering_wheel_angle</td>
<td>0.400024</td>
<td>-1.599976</td>
</tr>
<tr>
<td>parking_brake_status</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>high_beam_status</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>brake_pedal_status</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>headlamp_status</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>windshield_wiper_status</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>engine_speed</td>
<td>2738.0</td>
<td>1386.0</td>
</tr>
<tr>
<td>transmission_gear_position</td>
<td>4.0</td>
<td>6.0</td>
</tr>
</tbody>
</table>

TABLE 4: Comparing \(s_i \in S_{\text{high speed}}\) and \(s_j \in S_{\text{high speed}}\) with large ABAT\(e_{\text{score}}\)

We performed one more experiment to testify the above capability. In this experiment, we aggregated two sets \(S_{\text{gear} = 1}\) which is the set of all states where transmission gear position is 1 and \(S_{\text{gear} = 6}\) 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, \(S_{\text{gear} = 1}\) had 129 states and \(S_{\text{gear} = 6}\) 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.

Another broad category of attacks common in cyber-physical systems is FDIAs (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 \(\tau_{\text{anomaly}}\) is varied. Each line represents a specific value for \(\tau_{\text{anomaly}}\). 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

For evaluations, we generated 3 sets of sequences; sequence with $\Delta_{ia}$ greater than $\Delta_{sa}$ and $\Delta_{ss}$, and sequence with $\Delta_{sa}$ greater than $\Delta_{sa}$ and $\Delta_{ss}$. 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 $s_0 \rightarrow s_6$ and regenerating the sequence will produce a sequence with $s_0 \rightarrow s_6$ 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.

<table>
<thead>
<tr>
<th>End State</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>$s_0 \rightarrow s_1$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$s_0 \rightarrow s_5 \rightarrow s_4$</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$s_0 \rightarrow s_5$</td>
</tr>
<tr>
<td>$s_6$</td>
<td>$s_0 \rightarrow s_1 \rightarrow s_{11} \rightarrow s_6$</td>
</tr>
<tr>
<td>$s_{10}$</td>
<td>$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_{10}$</td>
</tr>
<tr>
<td>$s_{11}$</td>
<td>$s_0 \rightarrow s_1 \rightarrow s_{11}$</td>
</tr>
</tbody>
</table>

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 \( s_0 \to s_9 \) 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, \( r_i^t \), which is a straight forward function with complexity of \( O(1) \). The next task is to choose a \( v_i^t \) corresponding to it. Time complexity for calculating \( EU \) 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-liher 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


Sandeep Nair Narayanan Sandeep Nair Narayanan received B.Tech. degree from Cochin University of Science & Technology, M.Tech. degree from National Institute of Technology Karnataka, and Ph.D. degree from University of Maryland Baltimore County. He had worked for Oracle Corporation and is currently with Cisco Systems. His research interests are in network security, application security, and IoT security.

Anupam Joshi Anupam Joshi (Fellow, IEEE) received the B.Tech. degree from IIT Delhi, in 1989, and the master’s and Ph.D. degrees from Purdue University, in 1991 and 1993, respectively. He is currently the Oros Family Professor and the Chair of the Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County (UMBC). He is the Director of the UMBC’s Center for Cybersecurity. His research interests are in the broad area of networked computing and intelligent systems. His primary focus has been on data management and security/privacy in mobile/pervasive computing environments, and policy driven approaches to security and privacy. He has published over 250 technical papers with an h-index of 83, filed and been granted several patents, and has obtained research support from National Science Foundation (NSF), NASA, Defense Advanced Research Projects Agency (DARPA), US Dept of Defense (DoD), NIST, IBM, Microsoft, Qualcomm, Northrop Grumman, and Lockheed Martin amongst others. (Based on document published on 8 September 2020).

Ranjan Bose Ranjan Bose (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT) Kanpur, Kanpur, India, in 1992, and the M.S. and Ph.D. degrees in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1993 and 1995, respectively. From 1996 to 1997, he was with Alliance Semiconductor Inc., San Jose, CA, USA, as a Senior Design Engineer. Since November 1997, he has been with the Department of Electrical Engineering, IIT Delhi, New Delhi, India. He is currently the Director of the Indraprastha Institute of Information Technology Delhi. His current research interests include physical-layer security and IoT security.