ABSTRACT

Title of dissertation: A Framework for Detecting Anomalous Behaviors in Smart Cyber-Physical Systems

Sandeep Nair Narayanan,
Doctor of Philosophy, 2019

Dissertation directed by: Professor Anupam Joshi,
Computer Science & Electrical Engineering

This dissertation makes significant contributions to automatic, scalable, and data-driven approaches for securing smart cyber-physical systems (CPS). Smart CPS are increasingly embedded in our everyday life. Security incidents involving them are often high-profile because of their ability to control critical infrastructure. Stuxnet and the Ukrainian power-grid attack are some notorious attacks reported against CPS which impacted governmental programs to ordinary users. In addition to the deliberate attacks, device malfunction and human error can also result in incidents with grave consequences. Hence the detection and mitigation of abnormal behaviors resulting from security incidents is imperative for the trustworthiness and broader acceptance of smart cyber-physical systems. In this dissertation, we study the behavior of smart cyber-physical systems and develop techniques to abstract the typical behaviors in such systems using the data generated from their components and detect various abnormalities. Our initial research developed a knowledge-graph based approach which uses semantic technologies to infer complex contexts for de-

tecting a wide range of anomalies. We also propose an automatic behavioral abstraction technique, ABATe, which automatically learns their typical behavior by finding the latent "context" space using available operational data. The learned latent space is then used to discern anomalies. We evaluate our technique using two real-world datasets to demonstrate the multi-domain adaptability and efficacy of our approach. As a part of this dissertation, we also generated an automotive dataset to support future research in the related fields.

A FRAMEWORK FOR DETECTING ANOMALOUS BEHAVIORS IN SMART CYBER-PHYSICAL SYSTEMS

by

Sandeep Nair Narayanan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Anupam Joshi, Chair/Advisor
Professor Tim Finin
Dr. Nilanjan Banerjee
Professor Yelena Yesha
Dr. Filip Perich
Dr. Wenjia Li

Dedicated to my Parents,
Brother, Sister-in-law, Teachers,
&
my Wife
who encouraged me unconditionally.

Acknowledgments

I thank each and everyone who helped me for the successful completion of this dissertation work. First and foremost, I would like to express my sincere gratitude to my advisor Prof. Anupam Joshi for his valuable advice on research, professional, and personal matters during my Ph.D. I am indebted to him for helping me to develop as a better researcher and a better professional during my time at the University of Maryland Baltimore County.

I would also like to thank my dissertation committee members Prof. Tim Finin, Dr. Nilanjan Banerjee, Prof. Yelena Yesha, Dr. Filip Perich, and Dr. Wenjia Li for sparing their valuable time to help me in my research. I owe the Ebiquity lab at UMBC for the successful completion of my Ph.D. research and thank Ebiquity labs affiliated faculty, Dr. Karuna Joshi and Dr. Francis Ferraro for their help in various matters during my time here. I thankfully remember all the Ebiquity lab members including senior Ph.D. students (Abhay, Prajit, Jennifer, Claire, Lisa, and Mahbubur), my cohorts, alumni, graduate students, and undergraduate students with whom I had the pleasure to work with. All of them made me feel home with their support.

I am thankful to my Ph.D. cohort Sudip Mittal, Ankur Padia, and Nisha Pillai with whom I enjoyed my student life at UMBC. I also thank my previous roommates Anin Puthukkudy and Deepak Krishnankutty for helping me, understanding me, and involving me in various curricular and extra-curricular activities which made my life in the United States enjoyable and cheerful. During my Ph.D., I also had

the pleasure to work with several student organizations at UMBC like the ACM student chapter, Graduate Research Conference committee, and Graduate School Association where I helped organizing several events. I thankfully remember all the staff at the Department of Computer Science and Electrical Engineering at UMBC and International Education Services for making my UMBC experience enjoyable.

I also take this opportunity to acknowledge the value of several sources (research grants, gifts from the industry partners like Fujitsu Labs of America, IBM, etc., funds from the Oros Family Professorship, GSA, and many others) my advisor, Prof. Anupam Joshi used to support my research at UMBC. I can never forget the efforts from my family (especially my dad, mom, brother, sister-in-law, and in-laws) who encouraged me all through these years. I am indebted to my beautiful and lovely wife, Arya (Neenu), for her sincere love and understanding during my Ph.D. effort. I sincerely apologize to anyone I had missed out unknowingly in this acknowledgment. Finally, I thank God Almighty for giving me the opportunity and courage to finish my doctorate.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

As of 2018, the number of IoT (Internet of Things) devices exceed 23 billion[1] and is expected to increase at a compound annual gross rate of 21% in the coming years[2]. B2C (Business to Consumer) use cases of IoT's like Google Home and Amazon Alexa are well-known. Recently, B2B (Business to Business) applications that improve operational efficiency, real-time control, and visibility in production are also gaining popularity. According to IBM[3], most of the IoT devices are also Cyber-Physical Systems (CPS), a network of connected devices which directly interact with the physical world and physical processes.

According to the National Science Foundation (NSF), "Cyber-physical systems integrate sensing, computation, control and networking into physical objects and infrastructure, connecting them to the Internet and to each other."[4] Sensors, actuators, and control systems that are connected over a network constitute the major logical entities in a typical cyber-physical system. Sensors measure the physical environment in which they are deployed and push the measurements to specific control systems. The control systems might differ in their capabilities such that

[1] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
[2] https://www.ericsson.com/en/mobility-report/internet-of-things-forecast
[3] https://www.ibm.com/blogs/research/2018/01/designing-cyber-physical-systems/
[4] https://www.nsf.gov/news/special_reports/cyber-physical/''

some just store the sensed measurements while others analyze it, alert other components (or human in the loop), or use the information to adapt physical parameters using various actuators. For example, in autonomous automobile systems, a radar senses the presence of obstruction ahead of the automobile and a control system integrates this information with other related sensors to come up with a decision on whether to change lanes or apply brakes. CPS's find application in a wide variety of domains such as agriculture [47], energy management [44], environmental monitoring [72], medical monitoring [20], process control [71], smart manufacturing [79], smart transportation systems [83], and so forth. Many cyber-physical system instances like smart grids, autonomous automobile systems, robotic systems, automatic pilot avionics, medical monitoring, and process control systems play a significant role in the vision of smart cities as well.

The wide range of cyber-physical system applications and their ability to affect the masses make them a particularly lucrative target for hackers. Several reported attacks against CPS's affected entities ranging from governmental organizations and large corporate companies to ordinary users. Some of the attacks even have a direct impact on the economy of a nation. Consider the attack on Ukrainian smart grids [10]. It left a whole city without heat and electricity in the cold of December for many hours. Stuxnet which is dubbed as the world's first digital weapon [81], is another classic example of attacks targeted towards CPS's. It is often touted as an attempt to derail a nation's nuclear development program. Stuxnet infected the PLC's (Programmable Logic Controllers) of a nuclear plant and rapidly altered the speed of its isotope enrichment centrifuge which resulted in their premature physical

damage. Such undesirable activities were performed just once in a month to avoid detection by simple monitors. Stuxnet malware code was so sophisticated that it penetrated even the air-gapped network. It is believed that such an attack could only be masterminded by state-sponsored actors.

A CPS domain which demonstrated fast advancements is the automotive domain. Although automobiles with technologically advanced control systems are capable of providing efficient transportation and mitigating accidents, the potential attacks against them are alarming. The famous 2016 Jeep Hack [50] in which the researchers manipulated an unaltered moving vehicle on the road resulted in Chrysler calling back 1.4 million[5] vehicles. Mirai botnet attacks was another notorious example in which many smart home devices were added to the bot network by exploiting known vulnerabilities. Later this army of bots (300k devices at its peak[6]) was used to generate a Distributed Denial of Service (DDoS) attack on the scale of 250 Gbps[7].

The security of cyber-physical systems includes more than just defending against cyber-attacks. Device malfunctioning and human errors are among the other important security issues which must be addressed. Securing them against such problems is also vital for their wide-scale acceptance because, in many application domains like medical monitoring and autonomous navigation systems, they control critical systems and abnormalities can result in grave consequences. One such incident caused by a combination of device malfunction and human error is the

[5] https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/
[6] https://www.computerworlduk.com/galleries/security/timeline-of-mirai-internet-of-things-botn
[7] https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html

Air France flight 447 crash that resulted in more than 200 fatalities. Major causes of the accident[8] were attributed to the temporary failure of pitot probes caused by ice crystals, and the inability of pilots to comprehend the situation. Due to the icing, pitot tubes that measure airspeed during the flight malfunctioned and reported wrong values to the pilots eventually resulting in the aircraft's fatal crash. Another incident which was caused by device malfunction is the recent Lion Air flight crash[9] that resulted in 189 fatalities. According to preliminary investigations, an errant sensor triggered the MCAS (Maneuvering Characteristics Augmentation System), an autonomous safety system which avoids engine stalls (Engine stall is a situation in which there is not enough thrust on the wings to support the airplane's weight at a very high altitude. Engine stalls often result in a free fall of the flight). The general practice to recover from a stall is to point the flight's nose down and allow the flight to gain more airspeed. However, in the case of the Lion Air flight, the faulty sensor made the flight's nose down at a very low altitude causing the pilots to correct it by pulling the nose up manually. This process was repeated more than 20 times before the crash. It is pointed out that the emergency procedure in such a situation was to disable MCAS. As a result of another presumed similar incident with the same aircraft (Boeing 737 Max 8), many countries and airlines grounded[10] their Boeing 737 Max 8 aircraft's. This incident reveals that the detection and mitigation of abnormal activities are imperative for the trustworthiness and broader

[8] https://web.archive.org/web/20120711071354/http://www.bea.aero/docspa/2009/f-cp090601.en/pdf/f-cp090601.en.pdf

[9] https://www.insurancejournal.com/news/international/2018/11/30/510682.htm

[10] https://www.cnbc.com/2019/03/12/uk-has-grounded-all-boeing-737-max-aircraft.html

acceptance of cyber-physical systems.

One approach to secure systems against cyber-attacks is to add advanced security features. Layering on a complex security system is impossible in many cases due to a variety of constraints [61] including physical environment feedback, distributed control, real-time response, wide-scale geographic distribution, and multi-tiered characteristics. The area, power, and cost requirements also put constraints on the design of cyber-physical systems and make it difficult to have advanced software and hardware security stack on them. For example, the geographic distribution and remote locations of the components in a power grid CPS may result in reduced network connectivity. Mobile CPS's like health and fitness monitoring systems might be power constrained. Support for legacy systems is another constraint against adding advanced security features. CAN bus, a broadcast based legacy communication channel lacks even the most basic authentication and authorization techniques. This deficiency is exploited in many attacks on cars which range from simple driver distractions to complete remote takeover of an unaltered car [30, 49, 50]. However, moving away from CAN networks is often not permitted due to cost and compatibility constraints. Besides, the security features alone may not handle device malfunctioning and human errors as they cannot detect and mitigate anomalous behaviors. All these show the importance of anomaly detection based solutions in this domain.

1.1 Intuition & Problem Statement

An identifying feature that differentiates cyber-physical systems is their direct interaction with the physical environment. Such interactions constrain the operational behaviors of their components. An instantaneous increase of speed values from the speed sensor of a car, for instance, is suspicious because the laws of physics constrain such behaviors in normal conditions. Similarly, in the case of the Stuxnet attack, a sudden speed variation of the centrifuge should not have occurred because the domain behaviors were well understood. Operations of a cyber-physical system might also follow certain domain etiquette's. For instance, it is not normal to allow filling a tank above its specified capacity or to allow a car to be driven in the night with the headlights off. However, there are several challenges in capturing such normal domain behaviors.

1. A cyber-physical system may be a collection of components from several domains and the relationship between their components can be very complicated. Sometimes these relationships can even be non-intuitive. Hence, the manual extraction of relationships is a tedious task which may require experts from several domains to work together.

2. The number of components in a cyber-physical system can be high. In the automotive domain, the number of sensors is projected to be close to 200[11] per car. This makes it hard to list out all their relationships manually.

[11] http://www.newelectronics.co.uk/electronics-technology/automotive-sensors-market-is-booming/149323/

3. A cyber-physical system may behave in different ways when exposed to different environmental conditions. Extreme climatic and geographical conditions affecting the normal operational behaviors and sensor values of an automobile is such a case. As a result, the set of normal operational behaviors for the same cyber-physical system might differ which makes the task more difficult.

With the advent of communication technologies and smart cyber-physical system infrastructure, a large amount of normal operational data is available. This data encapsulates different complex behaviors that may exist in that specific domain. We hypothesize that a solution which ingests this normal operational data from a cyber-physical system and abstracts its common behaviors or contexts of events may be able to detect unusual behaviors specific to that system. In this dissertation research, we aim to develop such a detection strategy which can aid cyber-physical systems to mitigate abnormal behaviors, both attack and device malfunctioning scenarios.

1.2 Thesis Statement

"We can learn a latent space that abstracts a smart cyber-physical system's typical environment specific behaviors using the plethora of operational data generated by its different components like sensors, actuators, and control systems. The generated latent space can be used to develop more potent, scalable and domain-independent solutions to detect behavioral anomalies and attacks."

1.3 Anomaly Detection in Smart Cyber-Physical Systems

This dissertation focuses on learning a latent space which can be used to detect anomalous behaviors using the normal operational data from smart cyber-physical systems. Significant contributions in the dissertation are enumerated below.

1. Developed a domain-specific approach to identify abnormal behaviors in cars using a knowledge-graph based approach. In this research, we designed an extension to the IoT Lite Ontology such that it accommodates cyber-physical system specific features and updated its A-Boxes with entities from the automotive domain. We also developed a system which extracts local contexts directly from the raw bits on the CAN (Controller Area Network) bus. Unlike simple rule-based systems, in our system domain experts can write general behavioral descriptions using SWRL (Semantic Web Rule Language) and a reasoner infers complex contexts from the extracted local contexts to detect atypical behaviors.

2. Proposed a Hidden Markov Model (HMM) based system that can detect anomalous behaviors in a smart car environment. As a part of this research, we extracted data from the CAN bus of a car, discretized it, and modeled it to form a sequence of observations. We then solved the problem of detecting anomalous behaviors by considering it as a sequence anomaly detection problem. In our solution, after modeling the data, we learned the HMM parameters using the Baum-Welch algorithm. We then developed a posterior

probability based technique to detect anomalous activities in the sequence. A significant outcome from this research is a vindication of our hypothesis that normal behaviors could be learned from the operational data of cyber-physical systems (in this research, cars).

3. Our next major contribution is developing a domain-independent and scalable solution that can support the efficient detection of abnormal activities in any smart cyber-physical system environment. This novel technique considers the cyber-physical system as a black box generating data and tries to learn relationships between the data coming out of it. In this solution, we combine similar vectors from data using Euclidean distances and generate a fixed list of states that can directly identify point anomalies. Further, we developed a novel technique based on neural networks that generates a latent space where contextually similar states appear together while keeping the dissimilar states apart. By integrating these two techniques, we find context vectors in the latent space corresponding to each observed state.

4. We developed a more efficient and potent technique that uses context vectors in the newly generated latest space to associate a score for every event in the cyber-physical system under consideration. This newly generated score denotes the efficacy of that specific event in its context. We implemented the complete system using python and tensorflow architecture and evaluated our system using two real-world cyber-physical systems; Secure Water Treatment (SWaT) plant dataset and automotive dataset. We demonstrate the ability of

our technique to detect attacks using the SWaT dataset and context abstraction features with a new automotive dataset.

5. For evaluations, we used a real-world dataset extracted from a scaled down water treatment plant. However, to test the multi-domain applicability of our technique, we needed another dataset. A major time consuming and challenging engineering contribution is to develop a new automotive dataset rich with several sensors. We believe that this dataset will help the research community to experiment and try out newer techniques on real-world datasets than being constrained on synthetic datasets. To create the dataset, we identified and evaluated several techniques to collect data from automobiles like ELM 327 based chipsets, STN 1100 based chipsets, Arduino based solutions, OpenXC Platform, etc. We subsequently developed an OpenXC based solution to aggregate data from various sensors of an automobile. After data collection, we aggregated and characterized close to 1000 miles (or 25 hrs) of standard driving data which constitutes diverse driving conditions including long and short drives on highways, country-sides, mountainous terrains, and rainy weather.

1.4 Dissertation Organization

In chapter 2, we discuss some related domains of cyber-physical systems and how they differ from each other. Apart from the high profile attacks on cyber-physical systems this dissertation is motivated by growing prominence of car as a smart cyber-physical system and the attacks which were demonstrated against them.

Hence, this chapter also describes the car as a smart cyber-physical system, its attack surface and different attacks against them from the recent literature. Chapter 3 looks at deep dive into the research literature relating to smart cyber-physical system security, anomaly detection, and automotive security. One significant engineering contribution in this dissertation is the creation of an automotive dataset. Details about the data collection process and data characterization is detailed in chapter 4. While Chapter 5 details the knowledge-graph based approach for anomaly detection, chapters 6 and 7 details the automatic behavioral learning in cars and smart cyber-physical systems and the conclusion and future directions ensue in chapter 9.

Chapter 2: Background

Businesses benefit from the introduction of connected smart systems in their respective domains. According to an article published in Forbes [16] , Harley Davidson reduced the build to order cycle by a whopping 36 times by employing IoT in their manufacturing plant and Rolls-Royce increased the fuel efficiency of jet engines which results in a total savings of $250, 000 per plane per year. Hence several competing technologies emerged which have similar characteristics but have their own identifying features. Some of them are enumerated below

- **Internet of Things (IoT):** European Research Cluster defines the Internet of Things (IERC) as "A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network". In IoT's each "thing" can be a full system running a full operating system or a bluetooth beacon which simply transmits a fixed URL at constant intervals.

- **System of Systems (S0S):** System of Systems is a collection of task-oriented or dedicated systems that pool their resources and capabilities together to cre-

ate a new, more complex system which offers more functionality and performance than simply the sum of the constituent systems. Currently, systems of systems is a critical research discipline for which frames of reference, thought processes, quantitative analysis, tools, and design methods are incomplete [65]. SoS differs from IoT's because in SoS each component is a full system unlike in IoT's.

- **Industrial IoT (IIoT) / Industry 4.0:** IIoT is named differently in different countries. Some of the different names include *Smart Industry*, *Industry 4.0* (4.0 denotes fourth industrial revolution), *Smart Factories*, and *Advanced Manufacturing*. All of these generally stands for the use of various IoT technologies in manufacturing and industrial processes and depends on innovations in Information and Communication Technology (ICT), CPS, network communications, simulations, and so forth.

- **Cyber-Physical Systems:** Definition for CPS's from NSF states "Cyber-physical systems integrate sensing, computation, control and networking into physical objects and infrastructure, connecting them to the Internet and to each other.[1]". The differentiating feature of CPS's is their ability to monitor (and sometimes control) physical processes and physical activities.

[1] https://www.nsf.gov/news/special_reports/cyber-physical/

2.1 Cyber-Physical Systems

In this dissertation, we focus our research on smart Cyber-Physical Systems (CPS). As mentioned in the definition, it is an interconnected network of devices which interact with physical entities. In a typical CPS, there are 3 main entities; *Sensors* that sense the physical environments, *Control systems* that aggregate, process, and analyze the sensed physical measurements (may/may not generate actionable intelligence), and *Actuators* that can alter the physical state of its deployment according to instructions from the control system. According to CPSE[2] typical CPS may control and monitor various processes(physical or organizational), integrate solutions from multiple disciplines, handle human interactions, optimize its own performance, evolve (and adapt) according to the changed environments, involve a large-scale (even with a hierarchical structure), and so forth.

2.1.1 Applications

CPS domain is multi-disciplinary and finds application in several domains. A comprehensive literature review on various applications and research in the CPS domain is presented by [14]. Some of them are enumerated below.

- **Agriculture:** Several techniques are used for efficient food consumption, and food production capabilities. Some of them include precision agriculture, intelligent water management, etc.

[2] http://www.cpse-labs.eu/cps.php

- **Energy Management:** CPS are widely used to monitor, and transfer energy efficiently adapting itself to the user demands and availability. Smart grids is a well-known example.

- **Environmental Monitoring:** Monitoring environment is important for early detection of natural calamities like fire, flooding, Tsunami's, etc. CPS's are deployed to monitor and report such instances continuously.

- **Intelligent Transportation:** CPS's are used to manage complex traffic flow, safety, and situational awareness.

2.2 Car as a Cyber-Physical System

A popular cyber-physical system which acquired smart decision-making capability is cars. Automobiles as just moving parts, pulleys, and mechanical devices solely controlled by human drivers are antiquated long back, and they developed from being just mechanical devices into machines which can drive autonomously. A modern car has a large number of smart sub-systems (Fig. 2.1), and they make a lot of intelligent decisions every second. They can provide safe, efficient, and fast transportation in smart cities. Driver-assist features like anti-lock braking system, adaptive cruise control, and blindspot warning systems are becoming the de facto in many newer cars. For instance, the latest version of Accord from Honda Motor Company has *Honda Sensing*[3] (Honda's package of smart features like Collision Mitigation Braking System (CMBS), Road Departure Mitigation System (RDM),

[3] https://automobiles.honda.com/safety

Adaptive Cruise Control (ACC), and Lane Keeping Assist System (LKAS)) feature as a default, even for its base version. Current cutting-edge research in this domain focuses on autonomous driving in which a car can navigate you from one location to another without direct human intervention. Waymo[4] (an Alphabet company), Uber[5], GM motors[6], Tesla[7], etc. are some automotive giants who are innovating in this area. Waymo's fleet of autonomous cars has already driven over 4 million[8] combined miles on normal roads.

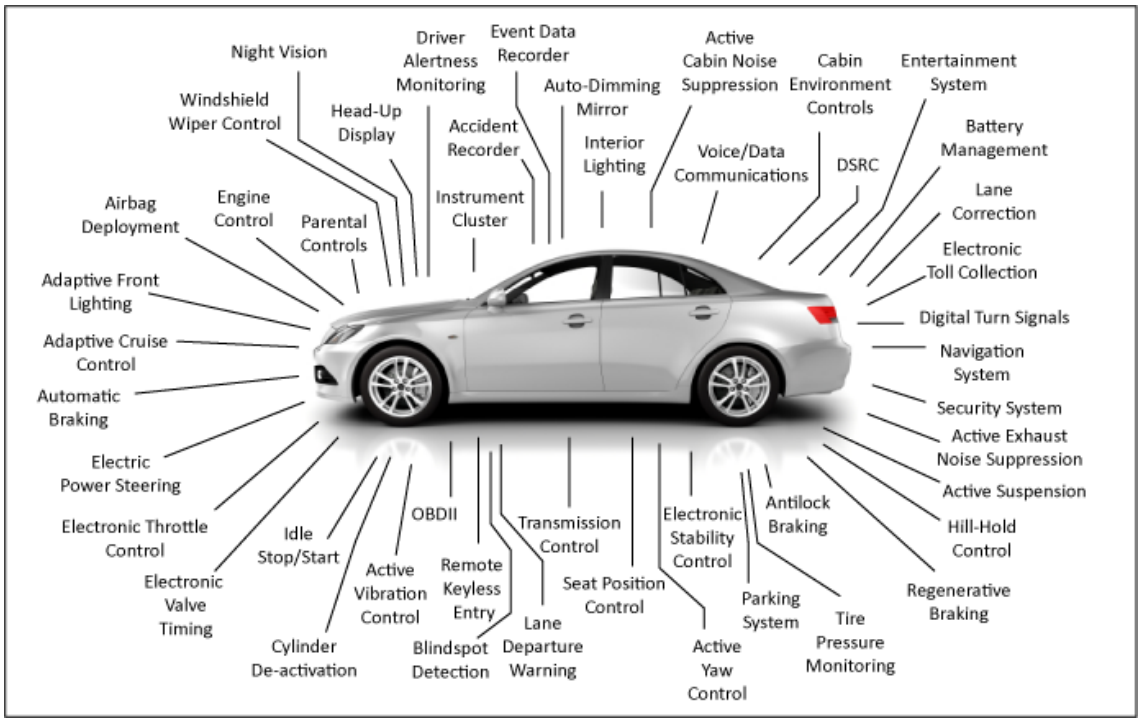


Figure 2.1: Modern Car Sub-Systems.

[4] https://waymo.com/
[5] https://www.uber.com/
[6] https://www.gm.com/
[7] https://www.tesla.com/
[8] https://waymo.com/ontheroad/

2.2.1 Modern Car Architecture

Advancements in microprocessor technologies resulted in the emergence of new and powerful ECU's (Electronic Control Units) in automobiles for making fast and smart decisions for many activities. Typically, an ECU will be connected to different sensors (eg. Acceleration sensor, Rain sensor, Ambient light sensor, Wheel speed sensor, RPM sensor, Vehicle Speed sensor, Oxygen sensor, temperature sensor, etc.) and actuators (eg. Instrument clusters, spark plugs, airbag inflators, etc.) using cables. Even though all automobiles have similar facilities, as of now, they don't have a standardized architecture. Different manufacturers like Honda, Toyota, BMW, Benz, Kia, and General Motors use different types of equipment, internal wiring architecture, and protocols. However, in general, we can see that all of them use one or more buses inside, and most of the subsystems and sensors are connected to them as shown in Fig. 2.2. Sub-systems need to meet specific constraints according to the task it is assigned to. For example, ABS (Anti-lock Braking System) need to meet strict time requirements. Otherwise, the vehicle won't stop at the required location. On the other hand, some sub-systems like entertainment subsystem need not meet the time requirement, but they need high volume data transfer. Different buses are chosen based on these requirements. Attributed to lack of standardization, some sensors are directly connected to the ECU, while others are connected to the common bus. Broadly each vehicle has at least a common bus and different smart subsystems.

2.2.1.1 Common Bus

Earlier vehicle manufacturers used a point-to-point wiring harness to connect all the components, and it became complicated due to an increase in the number of components. In 1986, Robert Bosch GMBH introduced a lightweight and low-cost serial bus protocol named Controller Area Network (CAN) for vehicular communication because none of the then existing protocols had the required characteristics. CAN was a broadcast based protocol in which all the devices are connected to the same bus, and all devices see all network communications. Error correction and priority maintenance were its important characteristics along with decreased complexity in the wiring harness. The newer CAN 2.0 specification was published in 1992 and became widely used by different manufacturers. The number of smart systems increased considerably over the past few decades. According to a report[9], the average number of such systems in a car increased from 24 in 2002 to 70 in 2013. To cope with this increase in the number of components and speed requirements of different subsystems, newer protocols and buses were introduced. Some of the protocols over CAN bus are ISO-TP protocol, CANopen, etc. A newer version of CAN named CAN-FD got introduced in 2011 which has flexible data rates. Other protocols introduced include PWM protocol from Ford, Keyword Protocol (KWP2000), VPW Protocol used in GM and Chrysler, LIN (Local Interconnect Network) protocol[10], MOST (Media Oriented System Transport) protocol[11], etc. Each of them has

[9] http://cvrr.ucsd.edu/ece156/AutomotiveSensors-Review-IEEESensors2008.pdf

[10] https://vector.com/vi_lin_spec_download_en.html

[11] https://www.mostcooperation.com/publications/specifications-organizational-procedures/request-download/mostspecificationpdf

specific characteristics. For example, LIN is a very cheap protocol to implement and use only a single wire bus. But it supports only up to 20Kbps. On the other hand, a version of MOST protocol supports up to 150 Mbps with the associated complexity of implementation. Another high-speed bus with a communication speed of up to 10Mbps is FlexRay[12]. Due to the diminishing support and increasing cost, newer vehicles are moving towards a newer protocol, automotive ethernet[13]. As of now, a common practice seen among manufacturers is to use different protocols and buses for different subsystems. Typically, MOST or FlexRay for high-end systems, CAN for mid-range systems and LIN for low-cost devices.

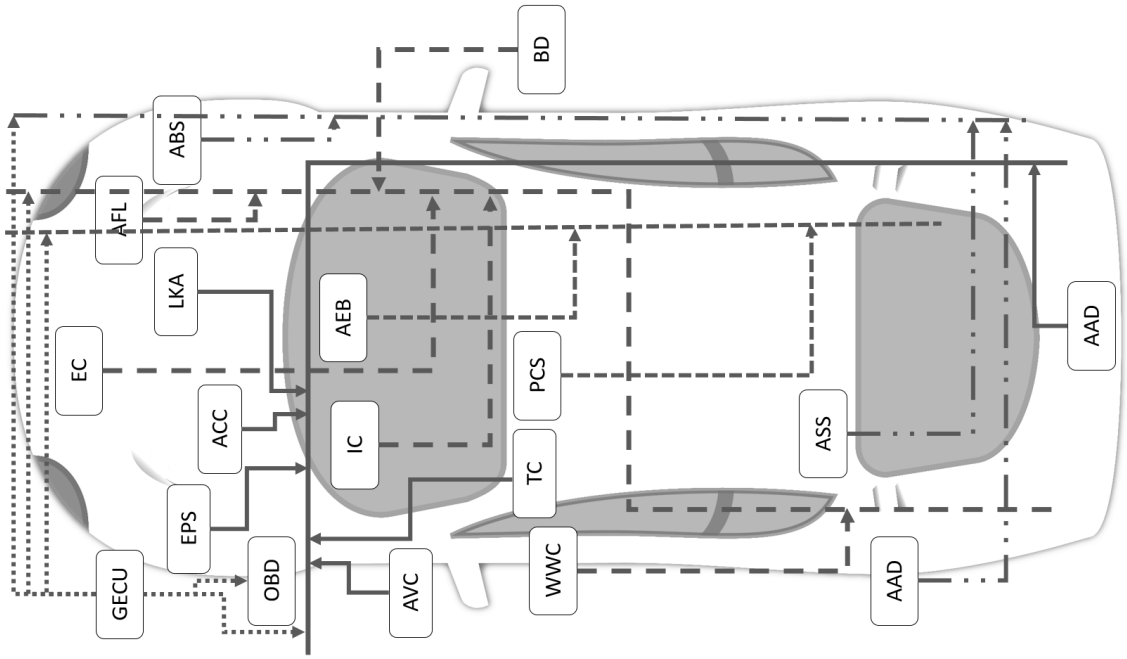
2.2.1.2 Smart Sub-Systems

More and more driver assist technologies are becoming the de-facto in a modern car. Some of the driver-assist technologies available in a car include anti-lock braking system, cruise control, lane-assist, power brake, power steering, central locking systems, etc. Technology is now taking the next step to move towards autonomous control in which the sub-systems will take over driving as a whole. Broadly we classify these subsystems into three categories.

- **Indicative Smart Sub-Systems:** Indicative systems identify certain states of the vehicle using their sensory inputs just to alert the driver. The smart blind-spot detection system is an example indicative system which detects objects in a car's blind-spot (the region around the vehicle which is not visible

[12] https://svn.ipd.kit.edu/nlrp/public/FlexRay/FlexRayProtocolSpecificationVersion3.0.1.pdf

[13] http://www.opensig.org/about/specifications/



EC	Engine Control	ABS	Anti-lock Braking System
GECU	Gateway ECU	AFL	Adaptive Front Lighting
LKA	Lane Keep Assist	BD	Blind-spot Detection
AEB	Autonomous Emergency Braking	ASS	Active Suspension Control
ACC	Adaptive Cruise Control	AAD	Active Aero-Dynamics
EPS	Electronic Power Steering	OBD	On-Board Diagnostics
PCS	Pre-Crash Safety	AVC	Adaptive Vibration Control
WWC	Windshield Wiper Control	TC	Transmission Control
IC	Instrument Cluster	SS	Security System

Bus 1
Bus 2
Bus 3
Bus 4

Figure 2.2: General Vehicle Architecture

to the driver using rear view mirrors) and alerts the user about it. It uses sensory inputs like radar, ultrasonic and vehicle speed sensor to detect an alien object presence. On detection, depending on the system, it alerts the user using a display, sound or haptic warning in the steering wheel or seats. Other examples of indicative systems include lane departure warning, tire pressure monitoring, etc.

- **Reactive Smart Sub-Systems:** Reactive smart systems, on the other hand,

take necessary actions to avoid potential bad incidents. A very common example of such a system is adaptive cruise control system which automatically adjusts the speed of a vehicle for the live road traffic. It uses a headway radar or lidar to detect traffic in front and utilize vehicle speed sensor, accelerator pedal position, and brake pedal position to identify the action to be taken for maintaining a safe distance on the road. It will then actuate throttle or brakes to adjust the speed of the vehicle automatically when the vehicle is in a cruise mode. Other example sub-systems include adaptive front lighting, parking systems, auto dimming mirrors, anti-lock braking systems, airbag deployment, etc.

- **Predictive Smart Sub-Systems:** In predictive smart systems, after identifying specific states of the vehicle, they anticipate certain actions from the driver. Instead of directly affecting the state of the car using actuators, they get the vehicle ready for such actions. A typical example of such a system is collision warning with brake support from Ford. In the event of an obstacle ahead and a potential collision, the system not only warns the user about the condition, it pre-charges the braking systems. This action will help to apply a full application of the brakes with a simple touch on the brake pedal and hence avoids an imminent collision. Precrash safety feature found in some automobiles is another example of such a system.

2.2.2 Attack Surfaces in a Modern Car

CAN bus is a broadcast based common bus which provides fast and efficient interconnection channel between different systems in a vehicle like Anti-Lock Braking System, Adaptive Cruise Control, Automatic Lane-Assist, and Adaptive Lighting Control. Due to lack of strong authentication primitives attached with CAN bus, any message on the CAN bus, in valid format, is a valid message. The attack surface describes all of the different points where an attacker could get into the system and where they could get data out[14] . The different actors in a vehicular set-up are driver, passenger, mechanic and external entities. We categorize the attack surface as *Internal Hardware Attack Surface, On-Vehicle Device Attack Surface, External Hardware Attack Surface*, and *Wireless Attack Surface*.

2.2.2.1 Internal Hardware Attack Surface

As described above, due to lack of authentication primitives any malignant internal hardware can be a potential attack surface. Normally the mechanic or any other external entities who can get physical access to the vehicle could exploit this attack surface. They can do it by reprogramming the different Electronic Control Units (ECUs) which are responsible for aggregating and controlling different associated systems present in a vehicle. For example, a different ECU might exist for different systems like Anti-lock Braking System, Adaptive Cruise Control, Adaptive Lighting Controls, etc. Another potential way to exploit this attack surface is by

[14]https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet

replacing the original ECU with a malignant one.

2.2.2.2 On-Vehicle Device Attack Surface

There are different devices which are accessible to driver and passenger actors. The driver has different controls like steering, brakes, steering mounted controls (audio systems, cruise controls, paddle shift gears), etc. The passengers also have access to systems like audio system, climate controls systems, etc. Most of these devices have only hardware controls. But nowadays they are replaced or added with software controls like touch screen displays, mobile phone apps etc. to control them. Some of the ways in which various actors can manipulate the vehicle are by attaching a USB drive to the audio system, or by introducing a malicious input into the onscreen displays or remote control units associated with different systems.

2.2.2.3 External Hardware Attack Surface

The trend of attaching external devices on to a vehicular system is becoming popular. Different insurance companies in the US like Progressive or wireless providers like Verizon have devices which could be attached to the OBD-II port of the vehicle, which is directly connected to the CAN bus. The insurance companies use this to analyze driver behavior so that they can give discounts, while some other devices help users to locate the vehicle in a parking lot, remote start the vehicle or track a vehicle in case of a theft. Most of these devices are driven by software which need not be devoid of bugs and could be exploited by malicious hackers. There are

some devices available in the market which just need to be placed on top of the CAN bus to tap the information from it. For example, CANCrocodile from WagenControl is a contact less CAN bus reader which allows to get data from CAN bus without interfering with the physical CAN bus.

2.2.2.4 Wireless Attack Surface

With the advent of wireless communication revolution, different communication technologies like Blue-tooth, Wi-Fi, Mobile networks, Satellite radio, GPS network, RF sensors, etc. are also made available in the vehicle. For example, Bluetooth is used to connect to the audio systems or OBD-II attached devices. Multiple Android and IOS apps are available now which can communicate with these devices. For connected vehicles, even the Internet is made available. Mobile applications are made available by many manufacturers for various smart cars which can be controlled using mobile devices. Another potential access point is through the RF devices used for central locking systems. All of these systems present a new potential attack surface the attackers can try to exploit.

2.2.3 Cyber-Attacks on Cars

An unwanted after-effect of automobiles becoming autonomous or semi-autonomous is the potential for attackers to confuse it or hack it. Researchers and hackers enlist various attacks possible on automobiles. We categorize these attacks under three different labels as described below.

2.2.3.1 Direct Physical Access Attacks

In such attacks, the attacker will have complete access to the vehicle, and he can modify the software and hardware in it at his will. A potential attacker can be your mechanic or a car wash person. Hoppe et al. [30, 31] demonstrated many such attack scenarios. In the first attack scenario, an arbitrary code added to ECU resulted in a zero-day attack in which the car windows opened when its speed exceeded 200 kmph (Kilometer per hour). In another attack, they demonstrated how airbag control systems could be removed from a car and could be masked using some injected code. Other demonstrations include attacks on gateway ECU and warning lights.

Research from Koscher et al. [38] evaluated a car and performed extensive lab and on-road tests to demonstrate various attacks. On-road tests were performed at a decommissioned airport runway with a lot of safety precautions. Researchers manually introduced packets into the vehicular system to perform the attacks. Among the different attacks performed some could be manually over-ridden while others could not be. It is not very surprising that many of the attacks can be done on a running car also. Attacks demonstrated include frequent activation of lock relays and windshield wipers, trunk popping, permanent horn activation, disabling window and key lock relays, temporary RPM increase, idle RPM increase, prevention of braking, uneven engagement of car brakes, falsified speedometer readings, radio volume increase, car alarm honking, engine killing, etc. Some of the performed attacks are innocuous but if used with malicious intent they can cause loss of reputation

and trust. For example, if the car engine stops suddenly the reliability aspect will be badly affected.

2.2.3.2 Indirect Physical Access Attacks

In indirect physical access, the attacker cannot modify the hardware but can introduce alien devices like CD's, USB's and OBD (On-Board Diagnostics) devices. Devices which are connected to the OBD port of a vehicle with advanced external connectivity is a new trend in automobiles. OBD port, being connected to the external CAN bus can fetch various information about the car including error codes. Most of these tools provide intelligent information about the car and are connected to the internet/cloud. Some of the popular devices include Automatic[15], Progressive Snapshot[16], Ford Reference VI[17], Verizon Hum[18], etc. and a wide range of corresponding mobile phone applications.

Miller and Valasek [49] demonstrated various attacks possible on cars using this OBD port (An OBD port is mandated by federal law in all cars in the United States from 1996. It is generally used for monitoring emissions statistics by motor vehicle departments, checking the health of a car by mechanics, etc.) of a car. They used ECOM cables to communicate with the CAN bus of two vehicles, Ford Escape, and Toyota Prius, and used tools like ECOMCat to read and write to the bus. Their research started with simple attacks like displaying false data on the dashboard, and simple DoS (Denial of Service) attacks by overwhelming the bus. Eventually, they

[15] https://www.automatic.com/
[16] https://www.progressive.com/auto/discounts/snapshot/
[17] https://shop.openxcplatform.com/ford-reference-vi.html
[18] https://www.hum.com/productstext

showed more advanced attacks like disabling brakes, applying brakes, killing the engine, etc. by introducing crafted packets to the CAN bus.

2.2.3.3 External Attacks

In such attacks, attackers will make use of the external open interfaces or software vulnerabilities on an unaltered/factory condition vehicle. Researchers Miller and Valasek [50] famously hacked into an unaltered Jeep vehicle and controlled it remotely. The attack reportedly resulted in a 1.4 million vehicle recall by Chrysler. The Jeep is an advanced connected vehicle which has mobile connectivity built in using the Sprint network in the US. Researchers reverse engineered and found that the D-Bus service (a software bus for interprocess communication and remote procedure calls) is running on the port 6667 and can be accessed anonymously. It will also allow anyone to run arbitrary code on the vehicle's head unit. Interestingly, they discovered that vulnerable vehicles can be found by just scanning the port 6667 on IP addresses starting with 21.0.0.0/8 and 25.0.0.0/8. The initial steps in the exploit chain include target identification (the IP address of the vehicle by scanning for port 6667) and exploitation of OMAP chip in the head unit using D-Bus service. To control non-CAN features like Radio, only these steps are necessary. However, in order to get control of the vehicle, the attacker modified the v850 micro-controller firmware. Once done, the hacker is in a position to introduce arbitrary CAN messages to the vehicle and hence can remotely control the vehicle. Despite Fail-Safe practices like shutting down the system in case of irregularity, the attackers were

able to perform malicious activities like killing the engine while running, locking and unlocking the car, activate the blinkers and windshield wipers, partially control steering wheel (exploiting parking assist features, but only at low speeds), etc. Several other reports claim about a crypto attack [75] on key-less entry feature on cars from popular makers including Volkswagen, Skoda, Volvo, etc.

Chapter 3: Literature Review

Anomaly detection is successfully used in computer security [41] , biological domain [46] , mechanical domain [8, 9] , financial domain [11] , unusual activity detection [39] , user behavior [5], etc. In an extensive survey, Chandola et al. [12] 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. [78] 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 [22, 66]. Another anomaly detection technique based on hierarchical HMM's is proposed by Zhang et al. [82]. However, Warrender et al. [78] reported significant performance overheads when HMM based techniques are compared with other techniques in longer sequences. Keogh et al. [37] 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 em-

ployed. Sabokruou et al. [70] used fully convolutional neural networks to detect abnormal regions in a video. O'shea et al. [62] applied recurrent neural network for anomaly detection on air radio networks. Laskov et al. [43] 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 [42], EXPOSE [73], HTM [2], etc. Black box analysis techniques [15] are also used to detect anomalies in large-scale systems, using externally available information.

3.1 Security in Cars

One of the core problems with the vehicular architecture is the presence of comparatively primitive bus technology, lacking cutting-edge security features, with far more sophisticated subsystems running on top. For example, CAN bus is a broadcast based bus in which all connected devices can see all the traffic, and there is no inbuilt concept of authentication or authorization. Any message on the wire is authenticated by default. This enables an attacker to introduce well-formed packets on to the wire to perform malicious activities. Research enlists two categories of techniques to secure the environment which are described in the following sections.

3.1.1 Attack Prevention Techniques

The first approach is to prevent attacks from happening. According to Wolfe et al. [80], a combination of hardware protection techniques, software protection

techniques and secure communication is required to achieve this. Newer protocols (varying in capabilities, speed, implementation overhead, etc.) including modification of existing protocols are proposed with this intent. LCAP [29] (Lightweight CAN Authentication Protocol) is one such protocol with minimum overhead and modification to the existing CAN network. The technique is based on a pre-shared key and magic number exchanged between the sender and receiver. At the beginning of a drive cycle, all devices perform an initial setup and will have a session key, HMAC Key and a channel initial magic number which are used for subsequent message transmissions. For data exchange, the sender will append the magic number to the message and encrypt it using session key. The receiver will authenticate the message using the magic number after decrypting it. However, this protocol will have the overhead for cryptographic calculations.

CANAuth [77] is another proposed protocol which is backward compatible. This protocol also depends on a pre-shared key stored in a tamper-proof location with each of the entities. A session key will be generated using the pre-shared key and it will be used for authenticating messages. In CANAuth, considering the hard constraints on time and message length, the authentication data is transmitted out-of-band using the CAN+ [84] protocol. Each authentication message will have a 32-bit counter value to prevent replay attacks and an 80-bit signature with this protocol.

LiBrA-CAN [28] is another proposed protocol to prevent attacks which uses CAN FD[1] (newer CAN protocol with flexible data rates). LiBrA-CAN presents two

[1] https://can-newsletter.org/assets/files/ttmedia/raw/e5740b7b5781b8960f55efcc2b93edf8.

new paradigms namely key splitting and MAC mixing for message authentication. The protocol can be used in a master Oriented flavor (a node with higher computational capabilities is used for authentication) which includes centralized, cumulative, and load balanced authentication schemes, or distributed flavor which includes two-stage authentication and multi-master authentication schemes. In comparison with other protocols, they have efficient forgery detection using MAC mixing and lesser authentication delays.

Apart from protocol specific drawbacks like time constraints, the requirement of pre-shared keys and higher computational power are some disadvantages of such techniques. Another major drawback for this class of techniques is that we can only protect future vehicles not the present ones on the road. Moreover, the automotive domain is a huge industry and it requires a long time to change. Koscher et al. [38] describes this stringent "operational and economic realities" in the automotive industry and explains the importance of a detection strategy for automobiles.

3.1.2 Attack Detection and Mitigation

The second strategy to make the automotive domain safe is attack detection and mitigation. Different companies have come up with Automotive Intrusion Detection Systems. Panasonic Corporation is developing an IDPS[2] (Intrusion Detection and Prevention System) which combines Host Intrusion Detection technology (which combines behavioral information), in-vehicle device-type intrusion detection tech-

pdf

[2] http://itsworldcongress2017.org/wp-content/uploads/2017/11/kishikawa_20171025.pdf

nology (it monitors and detects unauthorized commands in the in-vehicle network, both CAN-based and Ethernet-based), and cloud-type vehicle intrusion detection technology (detects intrusion by analyzing logs collected from different vehicles using machine learning). Karamba Security[3] is another company which develops tools like *Karamba Carwall* (it generates policies based on factory settings and embed them into ECU's to continuously validate actions at runtime) and *Karamba SafeCAN* (it authenticates and hardens in-car network communications helping ECU's to ignore commands from invalid ECU's or physical hacks). Escrypt [4] is yet another company which develops an IDPS for vehicles. Their products, *CycurIDS* and *CycurGUARD*, analyze data from multiple vehicles and detect potential intrusions. We can classify the basic research in this domain into two sub-categories based on if the technique uses a semantic understanding of the underlying data.

Statistics Oriented Techniques: Research enlists different techniques for attack detection and mitigation. One such research direction is by applying machine learning techniques on raw CAN messages without considering the semantics of the messages.

Muter et al. [53] proposed an entropy-based approach for anomaly detection in vehicles. In this technique, various information theoretic measures are used to calculate entropy. Entropy, in general, measures "how much coincidence" of a given data set or it represents the abstract representation of randomness. However, due to the restricted vehicular network specifications, the amount of randomness (or entropy) is lower. Hence in their approach, they consider an adaptation of the

[3] https://karambasecurity.com/products
[4] https://www.escrypt.com/en/solutions-overview

existing entropy-based intrusion detection techniques. To calculate entropy, they use 3 levels of data abstraction. Binary-level data abstraction consists of the raw ones and zeros. At this level, either a bit-wise classifier (which consider each bit as an event) or x-bitwise classifier (combination of x bits is considered as an event) can be used. The next level of abstraction is signal level in which an event is generated for every signal value of the CAN message. The assumption is that there are only fixed messages in a vehicular network unlike a general protocol like TCP or UDP. The final level of abstraction is Protocol level. CAN defines 12 fields for data frames in the base format. The classifier in this level generates an event for every field in the CAN message. The entropy (Eq. 3.1) and relative entropy (Eq. 3.2) is calculated using standard information theoretic techniques. Their technique was put against simulated attack scenarios like *increased frequency*, *message flooding*, etc. and showed that most attack scenarios caused variations in the calculated entropy and could be utilized for detecting anomalies.

$$H(X) = \sum_{x \in C_X} P(x) log \frac{1}{P(x)} \tag{3.1}$$

where,

C_x : *Set of classes for dataset* X ,

$P(x)$: *Probability of* x *in* X.

$$RelEnt(p/q) = \sum_{x \in C_X} p(x) log \frac{p(x)}{q(x)} \tag{3.2}$$

where,

C_x : *Set of classes for dataset* X ,

$p(x), q(x)$: *Probability distributions over* $x \in C_x$.

Taylor et al. [76] use LSTM's (Long Short Term Memory) to detect anomalies. LSTM's are used to learn long-term dependencies in sequences of data. In their research, they considered each bit on the CAN message as a feature for the LSTM and trained the network. Logically, once trained, the network will be able to predict the next CAN message given a previous sequence of CAN messages. In this technique, they trained a separate LSTM network for each CAN message. For detecting an anomaly, a scalar anomaly score is required. To generate this score, they use the binary loss function which is defined as

$$L(\hat{b}_k, bk) = -(b_k log(\hat{b}_k + \epsilon) + (1 - b_k) log(1 - \hat{b}_k + \epsilon)) \tag{3.3}$$

where,

$b_k : k^{th}$ *bit in the Message at step* i,

$\hat{b}_k : k^{th}$ *bit's predicted value by the network,*

ϵ : *a fixed value that caps the maximum loss.*

The loss function will have a low value for incorrect and middling predictions and very high value for confident and incorrect predictions. The final anomaly score is calculated by combining the bit losses over the entire sequence using various strategies like *Maximum bit loss* (maximum of all bit losses over the entire sequence) *Maximum word loss* (maximum in average bit loss over all words), *Window max* (maximum in average bit loss over words in a window), *Log window max* (log mean of average bit loss over words in a window), *Sequence mean* (mean bit loss over the complete sequence), etc. To detect anomalies, an empirically found threshold will be used on the calculated scalar value.

In another parallel research, Kang et al. [34, 35] used deep neural networks (DNN's) directly on the bit stream. The motivation behind using a DNN is its ability to model nonlinear relationships. In their training phase, features representing the statistical behavior of the CAN packet is extracted. For feature extraction efficiency, they use each bit in the DATA field (64 bits) of a CAN packet as features. To reduce the dimension, they propose to split the DATA field into mode information and value information depending on the CAN message ID and avoid using unwanted bits from training. Now using these extracted features, they train a deep neural network with input layer's size as the number of extracted features and output layer with 2 neurons. In between, the network will have a fixed number of neurons. The 2 output neurons represent an attack packet and a normal packet respectively. That is, if the first neuron is activated, it signifies that it is an attack packet and if the other neuron is activated, it signifies a normal packet. In comparison with a feed-forward network, they were able to get better performance on their evaluation with data

generated using Open Car Test-bed and Network Experiments (OCTANE) [21].

3.2 Anomaly Detection in Cyber-Physical Systems

Detecting attacks in cyber-physical systems is a challenging task. In the CPS domain, Gollmann et al. [26] 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. [51] describes the challenges and various techniques proposed. Jones et al. [33] 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. [40] 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.

A data-driven approach is proposed by Liu et al. [45]. They use spatio-temporal features and learn the system-wide patterns using a Restricted Boltzmann Machine (RBM). Goh et al. [24] used recurrent neural networks to detect anomalies in cyber-physical systems. They used an LSTM to model complex temporal sequences and predicted the next expected output. The deviation of the actual sensor data and the predicted data using CUSUM is then used to detect attacks. However, they were only able to apply their technique on phase 1 of the SWaT dataset [23] we use for the evaluation of *ABATe*.

Several works in the power systems CPS use state estimation [17,63] to detect

attacks. DAD from Adepu et al. [1] is another distributed attack detection technique for a water treatment plant using invariants. Invariants are simply mathematical relationships between different properties in the system. DAD uses the internal working knowledge of the system to generate invariants, and such invariants are used to detect attacks. A general limitation of this technique is the manual generation of invariants which is difficult. Mitchell et al. [52] 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 the statistical properties of the 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 [52]. 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. [24], many of the existing techniques are signature based. Hence, a direct comparison of our technique with others is difficult. Moreover, a comparison to some recent unsupervised techniques using the same dataset is also not viable because some of them [24] are restricted to specific phases of the SWaT dataset, or they use domain specific characteristics [1] in their techniques. Hence, in section 8, 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.

Chapter 4: Automotive Data Collection

As mentioned in chapter 2.2, car is a CPS which people use in their day to day life and research enlists several attacks 2.2.3 on cars. Detecting them is quite a big challenge. However, a sensor-rich dataset to support such research was not available. Some of the existing datasets were either [1], limited in the number of sensors or were meant for image processing based approaches[2]. Hence, in this dissertation research, we collected a rich dataset with data from 13 sensors, as described in table 4.1 from cars.

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

Table 4.1: Components in the Automotive Dataset

4.1 Data Collection Techniques

Several techniques are available to collect data from cars. Some of the devices which are economical include using chipsets like ELM 327, STN1100, Arduino

[1] http://openxcplatform.com/resources/traces.html
[2] https://github.com/udacity/self-driving-car/tree/master/datasets

with CAN-BUS shield, etc. (Figure 4.1)which captures raw CAN bus messages [57]. ELM 327 is a micro-controller which decodes the OBD(On-Board Diagnostics) interface of cars via UART. Many off-the shelf implementations are available which can communicate using USB, RS-232, Bluetooth or Wi-Fi. STN1100[3] is another chipset which is a similar multiprotocol UART interpreter for OBD-II protocols with better, stability, and performance. Yet another tool which can interface data from cars is using Arduino (an open-source electronic prototyping platform) based CAN bus shield from SparkFun using a microchip MCP25515 and MCP2551 CAN transceiver. Several mobile applications like Torque, OBD Car Doctor Pro, Engine Link, etc. help observe some data on a mobile device like a tablet or cell phone.

(a) ELM 327 based devices (b) STN 11XX IC based devices (c) Sparkfun CAN Bus Sheild

Figure 4.1: Data Collection Devices from Cars CAN Bus

There are several tools like OpenXC from Ford, Octane CAN bus sniffer [7], Komodo CAN bus sniffer[4], Vehicle Spy[5], SavvyCAN[6], O2OO Data logger[7], etc. that can be used for the collection and analysis of data from car and each tool has their own advantages. For example, Vehicle Spy helps us to view all the messages in

[3] https://www.scantool.net/stn1110/
[4] https://www.totalphase.com/products/komodo-canduo/
[5] http://store.intrepidcs.com/Vehicle-Spy-p/vspy-3-ent.htm
[6] http://www.savvycan.com/
[7] https://www.vanheusden.com/O2OO/

different CAN buses simultaneously. It also allows capturing and replaying different captured sessions on to the CAN bus. However, most of them show raw CAN messages(Figure 4.2) with raw CAN bus ID's which doesn't give any clear intuition on the semantics of different messages because the there is no standard CAN ID's available for cars except for the codes for testing vehicular emissions. In this research, we use OpenXC platform developed by Ford motor company.

```
01 10: [7E8 04 41 10 01 1E AA AA AA ]
01 44: [7E8 04 41 44 80 00 AA AA AA ]
01 04: [7E8 03 41 04 3F AA AA AA AA ]
01 06: [7E8 03 41 06 84 AA AA AA AA ]
01 07: [7E8 03 41 07 82 AA AA AA AA ]
01 0B: [7E8 03 41 0B 34 AA AA AA AA ]
01 0C: [7E8 04 41 0C 0C 62 AA AA AA ]
```

Fig. 1: Intercepted CAN Messages.

Figure 4.2: Raw CAN Messages from STN1100 based tools

4.2 OpenXC Platform

We used the OpenXC platform[8] for data collection from cars. It is a combination of open-source hardware and software that allows developers to unlock the rich vehicular data and help them to extend vehicles with custom applications. The software/hardware combination supports several hardware devices like Ford Reference VI, CrossChasm C5 devices, Cross Chasm C5 BT, OpenChasm C5 cellular, CrossChasm C5 BLE, DIY chipKIT-based VI, etc. The software API's from OpenXC is available in python and Android which helps developers a standard interface to analyze and visualize data from cars. The software also supports data to be exported

[8] http://openxcplatform.com/

to standard formats like JSON.

Figure 4.3: Ford Reference VI

In our data collection setup, we used the Ford Reference VI (Figure 4.3) as hardware and Android API as software for OpenXC. Several firmware's are available for the device and depends on the vehicle used and the dataset which need to be collected. Its also supports a configuration which helps decoding unknown/custom CAN ID's also. During data collection, we flashed VI using the expected firmware and used a nexus 7 tablet that uses the Android API's to dump data from the vehicular CAN bus in JSON format. The collected data is a stream of discrete JSON messages with each JSON will carrying information from a specific sensor available at that time. Figure 4.4 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.

Our first task is to convert the stream of JSON into a time sequence of data from all the sensors. For this preprocessing, we aggregated all the JSON messages during a very small time frame say one second and generated a vector which contains data from all the sensors. We can categorize two types of devices sending data on

```
{"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}
```


Figure 4.4: Automotive data from OpenXC VI

to the car's common bus. The first type sends data continuously at every time instant (Speed, RPM, etc.) and the second type sends only event data when its status change (eg. Door Sensor, Head Lamp status, etc.). While generating time vectors, if data is not available from a particular sensor at that time slot, the status of that sensor is copied from the previous slot because it is assumed that the status of that sensor didn't change in that time slot. In total, 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 4.2.

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 4.2: Automotive Dataset Characterization

Chapter 5: Knowledge-Graph Based Approach to Detect Anomalous Behaviors in Cars

Chapter 2 describes the car as a smart cyber-physical system and describes various attacks possible against them. In this chapter, we develop a semantic based approach to detect anomalous activities in a car using the data collected from them. The core idea is to aggregate local contexts available from the underlying data and uses them to detect a global context using semantic technologies. In research, context had been defined by multiple researchers. Among them, a popular definition is "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications.*" [19] In the vehicular domain the entities include hardware devices like sensors, control units, actuators, wireless devices, etc. and the different information which can characterize the situation include the sensor measurements, actuator states, etc. Some of the sensory measurements include speed, acceleration, location of the vehicle, and distance proximity of the vehicle from another object, and the states of actuators can include state of headlights is "on", state of wipers is "off", etc.

Semantic contextual inference can offer more precision and maintainability

over other methods in such situations because they abstracts human intelligence. In our approach, we use various semantic web technologies like Ontologies and RDF represent each of its components separately and use a reasoner to identify any inconsistencies. The same context could be determined from different sets of sensors. Such redundancies could also be utilized to verify the context of the vehicle or otherwise to detect anomalies. For example, we can detect that the vehicle is moving from the speed sensors or by the combination of different other sensors like engine rpm and gear position of the vehicle. If both of them leads to the inference that the vehicle is moving, then it is a normal scenario. But if one infers the vehicle is moving, and the other set contradicts, it represents a potential anomaly. In our system, we collect streaming data from a vehicle's CAN bus and use this data against a set of SWRL rules[1]. These rules are developed from historical system data can be used to extract vehicular context. We discuss a few use-cases where we derive vehicular context using these SWRL rules and use them to detect anomalous states.

5.1 Approach

A typical vehicle has various sensors and actuators communicating over a common communication channel. Due to the primary focus on efficiency and simplicity, when the protocols were developed, it lacked required authentication features. As a result, any message available on the CAN bus is considered to be a valid instruction, and the receiving control units and actuators have no way to distinguish it from a malicious instruction. We try to aggregate and extract context from these different

[1] https://www.w3.org/Submission/SWRL/

data exchanges to distinguish normal data flow from a malicious data flow.

In a vehicle, the communication happens over the common CAN bus. We can tap the data flowing on the bus using the OBD-II (On-Board Diagnostic) port which is mandatory on all the vehicles in the US from late 1990's. The data flowing on the bus can use multiple protocols like ISO 15765-4 (CAN), ISO 11898 (raw CAN), SAE J1939, etc. Apart from this heterogeneity, the amount of data available on the common communication medium also prompted us to propose a multi-tiered mechanism to extract context. The different layers are depicted in Figure 5.1. The Local Context Detection (LCD) layer will convert the crude and heterogeneous data into a higher plane for further context aggregation. The Cross Component Context Inferencing Engine (C3IE) aggregates the different local contexts from LCD to infer overall state of the system. We also propose the use of a Rule Mining Engine to extract knowledge from historical data to support the inferencing process. Each of the components are described in detail below.

5.1.1 Local Context Detection

The data from the OBD-II port is a stream of bits which is first converted into valid CAN messages. The Local Context Detection (LCD) layer's first job is to extract such valid messages from the stream of bytes. The CAN messages can contain raw sensor measurements like temperature, speed, acceleration, throttle pedal position, etc. In order to make the inference process faster and meaningful, the LCD layer again processes these sensor messages into a higher context plane. For

example, the speed sensor emits the raw speed measurements at regular intervals. Instead of introducing the raw values as it is into the knowledge base, the LCD layer will process them and extracts more useful context like "high speed", "sudden acceleration", "normal speed", etc. and associates them with their respective entities in the Ontology.

Another important reason for having the LCD layer is to ensure portability of our system to different vehicles. Since the communication channels are not completely standardized and different manufacturers still use proprietary protocols, the LCD layer allows for the reuse of same rules and inference procedures. The main functions of this layer include conversion of raw bits into CAN messages and aggregation of similar CAN messages to generate more meaningful local context. Once the local context is inferred, the LCD layer pushes it into the C3IE's knowledge base.

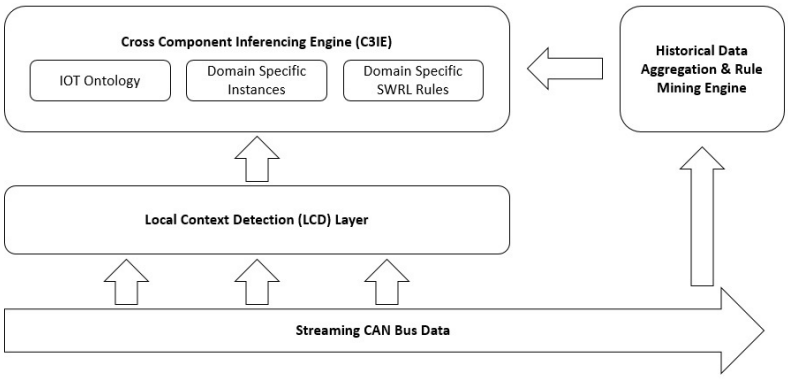


Figure 5.1: Overall System Architecture.

5.1.2 Cross Component Context Inference Engine

The LCD layer extracts the local contexts from different sensors and actuators from the common communication bus and delivers them to the Cross Component Context Inference Engine (C3IE). The four different components in C3IE are IoT Ontology which captures the logical semantics between different components, Domain Instances which are the specific instantiations of entities in the Ontology, Domain specific SWRL rules which captures the relationships between different components and a Reasoner which infers the current state of the whole system. The different components are explained in detail in the following subsections.

5.1.2.1 IoT Ontology

One of the benefits of developing an Ontology is knowledge reuse. Hence instead of developing a new ontology, we re-use the IoT-Lite [2] meta-Ontology, which is a light weight version of SSN [3] (Semantic Sensor Network) ontology. In order to match our specific use-case, we extend it by adding some properties. IoT-lite ontology is developed so as to allow cheaper processing time while querying it. It classifies the IoT devices as Sensing devices, Tag devices, and Actuator devices. Each of the devices is associated with an attribute which is a measurable quantity, and one or more devices are associated with it. We extend this ontology by associating object properties like "senseAttribute" and "affectsAttribute" to the devices such that we can keep track of the devices which are related to different attributes of

[2]https://www.w3.org/Submission/2015/SUBM-iot-lite-20151126/
[3]https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

the system. Also, we add a "hasStateValue" data property to the attributes of the system. A simplified view of the Ontology is presented in figure 5.2.

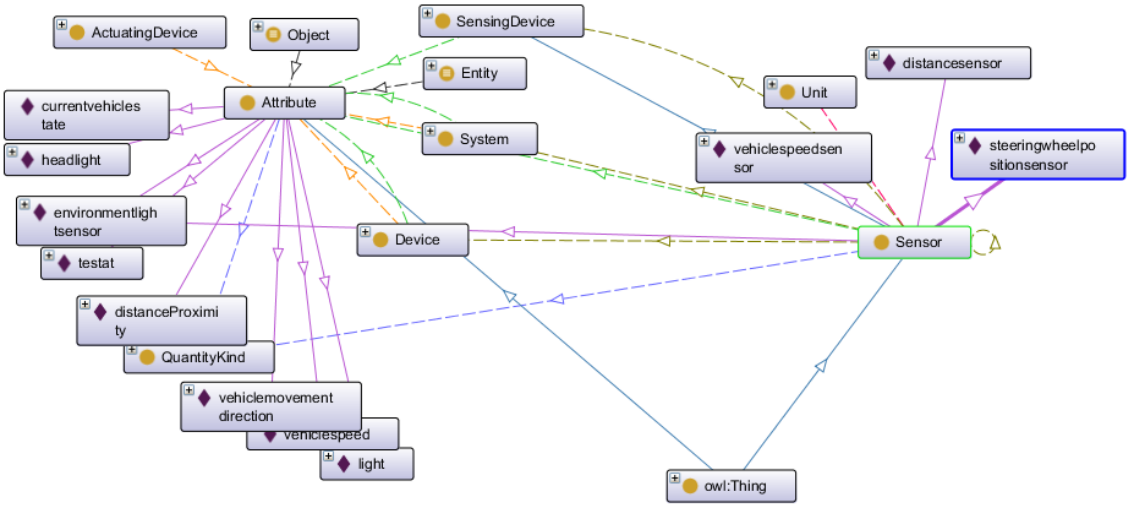


Figure 5.2: Simplified view of the Ontology used.

5.1.2.2 Domain Instances & Domain Specific SWRL rules

This is the part which customizes the ontology with domain specific knowledge. In the domain of vehicles, the different sensors may include "vehiclespeedsensor", "environmentlightsensor", "distacesensor" (which may be acoustic based, radar based or image based) and actuators like brake, accelerator, headlights, and step-up motors. Now we need to add domain specific rules for the reasoner to fuse local context information from different sensors, actuators, etc. and generate a global context. We use Semantic Web Rule Language (SWRL) to represent such rules in the Ontology. Some of the sample SWRL rules are specified in section 5.2.

5.1.2.3 Reasoner

In our Ontology, we define a special attribute called "currentsystemstate", which can have two values "normal" and "anomalous". It should be noted that our proposed technique is not only supposed to detect activities of malicious attackers, it also detects any inconsistent, dangerous or unexpected behavior of the system. For example, a sudden large change in the steering wheel angle, while the vehicle is moving at fast speed ranges is a dangerous behavior, caused by a malicious attacker or not. Sometimes it can be caused by a faulty component, but that doesn't stop the vehicle to be in a dangerous situation. Hence our proposed technique detects not only an attack state, but all dangerous or unsafe states. In the vehicular domain, we cannot use a static reasoner since the facts keep on changing. Hence we need to use reasoners like C-SPARQL [3, 4], IMaRS [27](Incremental Materialization for RDF Streams), TrOWL [64], C-SPARQL on S4 [6] or Streaming SPQRQL [6] which can work on the stream of data rather than on static set of facts.

5.1.3 Historic Data Aggregation & Rule mining

The presence of domain knowledge stored as SWRL rules is an important component of our system. The better information which could be captured with them, the better performance the overall system will have. Hence, we introduce the rule mining component, which uses statistical and machine learning techniques which generate these rules automatically. Once generated, the rules are ran against the already present rules and are tested for consistency before being added into the

SWRL rules.

5.2 Usecase Scenarios

Context-aware computing has been successfully deployed for multiple application domains. Here we try to apply it on the vehicular sensors in order to detect anomalous activities. As a part of analysis, we have collected live data from cars using CAN Bus shield and analyzed the possibility of converting them in to context in a higher plane. To create a meaningful test scenario, we consider a subset of sensors and actuators deployed in a real vehicle. Then we show the different instantiations in our Ontology, and some of the SWRL rules developed. The different sensors in the subset include "vehiclespeedsensor" which reports speed of the vehicle at regular intervals, "environmentlightsensor" which report intensity of ambient light around the vehicle, "distancesensor" which report the distance of the vehicle from an obstacle and "steeringanglesensor" which report current angle of the steering wheel from straight position. The actuators include "brake", "accelerometer" and "headlight". In the Ontology each sensor is associated with an attribute of the system, in this case a car. For example, the distancesensor will senseAttribute distance.

LCD layer will pick up raw CAN messages and interpret them to corresponding sensor values. Now the vehiclespeedsensor will be sending current speed of the vehicle in quick successions. But LCD layer will generate a higher level context for speed like, speed is "low", "high", "average" etc. Another example which depicts the functionality of the LCD layer is that different vehicles have different sensors for the

distance sensor. Some may have radar sensors while some others may have acoustic sensors. But LCD layer will convert these different values to a common distance value. LCD layer will continuously monitor the data flowing on the common bus and generate higher level contexts. When it detects a change in value of any of the local contexts for sensors, it will remove the corresponding local context of the sensor from the knowledge base and inserts new local context information. To insert new context, it will check the Ontology to infer that the corresponding sensor will affect a specific attribute of the system and it will update it accordingly. This will allow the reasoner to work on the attributes of the system directly. We will now describe SWRL rules which can be used to detect some abnormal behaviors.

```
hasComponent (vehicle, distancesensor)^ hasComponent (vehicle,
    vehiclespeedsensor) ^ hasStateValue (distanceProximity, "low") ^
    hasStateValue (vehiclespeed, "high") ==> hasStateValue
    (currentvehiclestate, "Anomalous Collision")}
```

In the above SWRL rule, it will first check if the vehicle has sensor components distance sensor and vehiclespeed sensor. This is added so that the rules will be generic for different vehicles. Some of them may or may not have a specific sensor. If they are available, the attributes associated with them will be properly updated. Hence in that case if the distance proximity has value “low” and vehiclespeed has value “high”, it implies an anomalous situation.

```
hasComponent (vehicle, environmentlightsensor) ^ hasComponent (vehicle,
    vehiclespeedsensor) ^ hasComponent (vehicle, headlight) ^
```

```
    hasStateValue (environmentlightsensor, "low") ^ hasStateValue
    (headlight, "off") ^ hasStateValue (vehiclespeed, "high") ==>
    hasStateValue (currentvehiclestate, "Anomalous No Light")}
```

In this example, in the presence of environmentlightsensor, vehiclespeedsensor and headlight actuator, if the ambient light is low and vehicle speed is high and headlight actuator state is off, then it represents an anomalous situation.

```
hasComponent (vehicle, steeringwheelpositionsensor) ^ hasComponent
    (vehicle, vehiclespeedsensor) ^ hasStateValue
    (vehiclemovementdirection, "high") ^ hasStateValue (vehiclespeed,
    "high") ==> hasStateValue (currentvehiclestate, "Anomalous Sudden
    Direction Change")
```

This is another example in which a sudden change in the angular position of the steering when speed is high in the presence of steeringwheelpositionsensor and vehiclespeedsensor inferring anomaly.

5.3 Discussion & Lessons Learned

In this chapter, we proposed a knowlege-graph based approach to detect anomalies in a smart cyber-physical system environment taking automotive domain as an example. From the usecase scenarios, it is evident that such an approach will be better than a simple rule based system in which every rule need to be mentioned. In our technique, experts provide abstract rules and our technique infers more com-

plex rules and situations from them using reasoning. Other important benefits of using this technique are lesser number of false positives and the explainability of decisions made because our system's decisions can be tracked to the knowledge-graph. In many learning based systems short term dependencies can be easily determined but the longer term influences like an event happening every one month are harder to be learned. They are better to be taught. Our system being taught such long-term as well as short-term dependencies can be incorporated.

Even though the cost of knowledge-graph reasoning is very high with a larger set of components, with powerful machines, fast communication channels, and cloud enabled technologies even larger systems could be managed in theory. Economical factors might affect the practical feasibility of such an approach. Moreover, the reliability and completeness of such a system is completely dependent on the closure of knowledge in the knowledge graph. This is a major weakness of this approach. With larger number and variety of components in a system, several complicated correlations exists. Even additional insights exists in the system which may not be intuitive. This makes other learning based techniques to be used in conjunction with such a system. Such techniques are discussed in the later chapters.

Chapter 6: Learning Constrained Behaviors in Cars

Chapter 2 describes the peculiarities of a typical cyber-physcial system. It interacts with the physical environment and may alter their current or future states. As a result, the operations of a CPS would be constrained by the various laws of physics and domain norms. For example, a car cannot alter its state from speed 60 mph to 0 mph in a second because of various laws of physics governing it. Similarly, in a normal situation nobody will drive a car with its door open (if the car has a door). In this section, we describe a Hidden Markov Model (HMM) based approach to capture such domain behaviors. We captured data from some components of a car and modeled it to fit an HMM model in this approach.

We try to convert the problem of detecting abnormal states in a vehicle into a data analytic problem. In this work, we first collected data from different vehicles and formulated the problem into a data analytic problem. Then we used HMM to create a model. Once the model is generated, we use it to predict any unsafe or anomalous states from the data flowing on the CAN bus. We evaluated our system by generating multiple anomalous scenarios by logically modifying the collected real data. We envision our system to be implemented on a portable programmable chip and connected to the CAN bus as an additional device. The device will monitor the

data on it, detect anomalous behaviors and generate alerts as required.

6.1 System Architecture

Newer protocols require significant modifications to ECU and sensor architecture. For future cars, to accommodate these changes would cause significant increase in manufacturing cost. But for existing cars, modifying the existing components would be hard and economically impractical. Since it is possible add third party gadgets, even to older cars, we believe that it is extremely important to make the in-vehicular network safe by detecting and possibly mitigating potential attacks. Hence we envision a technique which is applicable to older and newer vehicles at the same time with minimum modification to the existing architecture. We consider the stream of messages exchanged between different components of a car: Engine Control Module, Electronic Brake control module, Transmission Control Module, Body Control Module, Telematics, Radio, etc., forms a sequence of events and we formulate these sequence of events as a Machine Learning problem where we predict if the state of the vehicle is normal or abnormal. To accomplish it, we follow the following steps.

- Data Collection Phase: This is the first step in which the stream of CAN bus data is collected from real vehicles for analysis. We can employ the OBD-II port present in most vehicles for this purpose. Detailed discussion on data collection can be found in Section 6.1.1.

- Model Generation Phase: In this phase, we analyze the collected data and

generate a model. Since Hidden Markov Models (HMM) can abstract the time series data, we use them to model this scenario. Fitting the current scenario to HMM model is described in detail in Section 6.1.2.

- Anomaly Detection Phase: This is the final phase in which we detect anomalous behaviors in the vehicle using the generated model using posterior probabilities. It is described in Section 6.1.3.

6.1.1 Data Collection

The first step is to collect data from the CAN bus. CAN bus is a broadcast bus on which multiple devices are connected. When a device wants to communicate with other components connected to the bus, it will broadcast a message on to the bus with a specific message ID. As shown in Figure 6.1, each CAN message will have a specific Message ID(part before semi-column) and message data (part inside square bracket). While sending the message, each device will be identified by the Message ID alone.

```
01 10: [7E8 04 41 10 01 1E AA AA AA ]
01 44: [7E8 04 41 44 80 00 AA AA AA ]
01 04: [7E8 03 41 04 3F AA AA AA AA ]
01 06: [7E8 03 41 06 84 AA AA AA AA ]
01 07: [7E8 03 41 07 82 AA AA AA AA ]
01 0B: [7E8 03 41 0B 34 AA AA AA AA ]
01 0C: [7E8 04 41 0C 0C 62 AA AA AA ]
```

Figure 6.1: CAN Message

It should be noted that the OBD port, which is mandatory in many countries is also connected to the CAN bus in order to collect diagnostic information. Hence we

can essentially attach a device on to the OBD-II port and extract data for analysis. There are multiple tools like OBDLink Mx, Blue driver, CAN-BUS Shield with Arduino board and ELM 327 clone devices which can be attached to OBD port to extract raw messages broad-casted over it.

For data collection for this work, we used STN1100 based OBDLink MX. STN1100 is a multi protocol OBD to UART interpreter integrated circuit. It has a 16 bit processor with inbuilt flash memory and a RAM. It supports the complete AT command set (Command set for ELM 327 based chip-set) along with a new set of ST commands. It supports different protocols like ISO 15765-4 (CAN), ISO 11898 (raw CAN) and SAE J1939 (heavy vehicles). Other selected features include voltage input for battery monitoring and automatic protocol detection. We used OBDWiz, a tool which connects with OBDLink MX, to interface with the vehicles OBD port. During data collection, we set the "STMA" command, which will extract all the data flowing over the CAN bus. We collected data from vehicle from different manufacturers which include "Honda", "Toyota" and "Chevrolet".

We faced some practical limitations for collecting the data. Many of these vehicle manufacturers have different mechanisms which hinder the direct collection of data from the CAN bus. Some of the techniques include using multiple CAN buses which are guarded by different gateways. These gateways can be unlocked only by specific tools. But these simple techniques won't stop a malicious attacker to crack into it. We were able to collect the information from sensors like Vehicle speed, load, engine coolant temperature, Engine RPM, Intake air temperature, Absolute throttle position and O2 voltage using the tools mentioned. Because of the limitations and

difficulty in the data collection process, we moved to OpenXC platform for further data collection as described in Chapter 4

6.1.2 Model Generation

The second step in our approach is to analyze the collected data to develop a model which can identify anomalous states. In this project, we try to use Hidden Markov Models (HMM) to create a model. The intuition behind using this model is described below. We consider the movement of a vehicle is nothing but a sequence of states which are dependent on the previous state, like the Markov's processes. For example, consider the sequence of activities from T_1 to T_{12} as shown in Figure 6.2. At T_1 speed is zero and the Door is open. At T_2 the door is closed and it starts moving. The car gathers speed gradually till T_6. But at T_7 there is a sudden jump of 85 miles per hour making the speed to 100 mph. At T_8 the speed of car is 200 miles per hour and the door is open. We can clearly see that the probability of a state change from T_6 to T_7 and T_7 to T_8 are very low. Hence from these sequences, we can detect anomalous behaviors. We used HMM's to create a model since they provide powerful abstractions to predict time series data. To create a HMM model, we generate two set of probabilities, Transition probabilities and Emission probabilities. Transition probability controls how a new state, lets say "S(t)", is chosen from a current state "S(t-1)". The emission probability is probability that a specific set of observations will be generated given current hidden state "S(t)". During model generation we try to estimate these probabilities using the given data set.

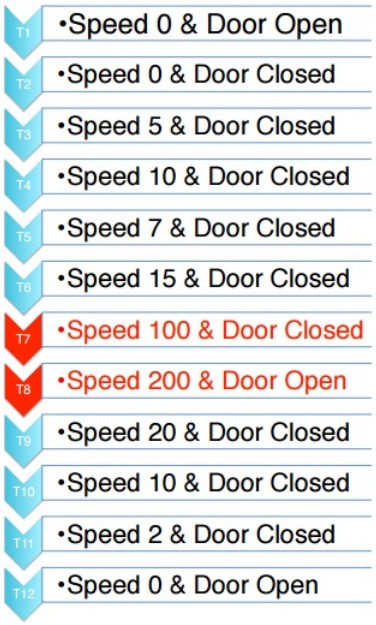


Figure 6.2: Sample Car Event Time-line

The first challenge for model generation is how to convert the collected data into a series of observations. The data collected in the first phase contain data from multiple ECU's. Instead of training the model with absolute values, we chose to use gradients for each observation, since the change in observations alters the state of the vehicle. For example in case of speed, instead of using actual speed, we find the speed gradients and train our system for it. The next challenge is on how to accommodate multiple observations as a single vector. We have different type of sensors in a vehicular system. Some of them will push data on to CAN bus at regular intervals like speed and RPM. On the other hand, there are some other observations which are pushed on to the system only when they are required like door sensors in some vehicles.

In our model, we create a vector containing different inputs from different

systems. Each vector will then represent a single observation, and our system will be trained for each of those observations. We generate observation vectors O_t for each set of values from different components available. We define the sequence of observations, $O = O_1, O_2, ...O_n$ where O_t is a vector. Each observation vector $O_t = \{v_{k,1}, v_{k,2}, ...v_{k,n}\}$ where $v_{k,i}$ is the value of the i^{th} component at time k. For example Speed is 20 mph, RPM is 3000, State of door is closed etc. are modeled as a single vector. During implementation, we interpret the different values from particular slots in the CAN message and convert into decimal values from Hex values before using them to train our model. To generate the HMM model we used HMM toolkit in Matlab. The generated sequence of observations is used to train the model using the "hmmtrain" function. We chose to use the Baum-Welch algorithm for training which will generate Transition and Emission Probabilities corresponding to test sequences.

6.1.3 Anomaly Detection

In this phase, we use the generated model to detect anomalies. By anomaly we mean a sudden deviation in the behavior of vehicle interpreted from data on CAN bus. As we described earlier, we are not only detecting attack states, but also any unsafe or anomalous states. For example, even though it is not caused by an attacker, opening of door at 200 mph is unsafe and hence we flag it. To detect unsafe states, we first convert the values from different components into sequence of observation vector in the same way as mentioned above. We then use a

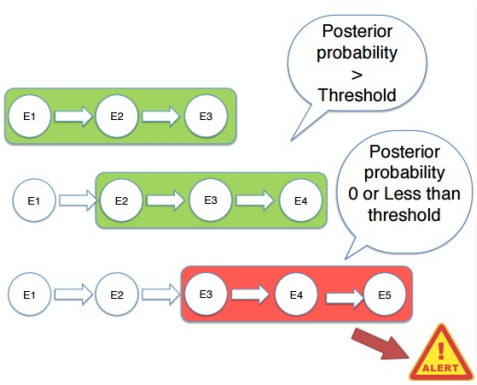


Figure 6.3: Sliding Window For Anomaly Detection

sliding window of "m" previous observations, $O_{window} = \{O_1, O_2, ...O_m\}$ as shown in Figure 6.3 to detect the presence of any anomalies. The sliding window moves every time a new observation is available. One of the operations which we can do with HMM is to detect the posterior probability of a given sequence. In this case, once the sliding window is determined, we use all observations in that window and determine the posterior probability of that sequence. In our case, each of the observation would be a vector of different sensor values. It will generate a set of probabilities corresponding to each observation. If the probability of any such sequence is below a threshold, based on the generated model, it implies that getting that observation in that sequence is very low and hence we identify it as an anomalous state based on our generated model.

We implemented anomaly detection using the matlab tool box. The anomaly detection module has the model as its first input. In our implementation, the input stream from the CAN bus is fed to this module. It will convert it into a sequence of observation using the same procedure we had used during model generation phase. Now when new observations are available, the module will pick up "m" previous

observations from the sliding window and use "hmmdecode" from matlab to find the posterior probability for the sequence in the window. The module will now generate an alert, if the probability of any observation in the sequence is going below a set threshold value.

6.2 Evaluation & Results

For our evaluation, we need to verify that no alerts are generated during normal conditions and alerts are generated during unsafe conditions. To test the normal conditions, we split the collected data into two parts. The first part is used for training the model and the second part is used to verify if the model generates any false positives. For evaluating the system to detect unsafe states, we hand crafted different scenarios by injecting unsafe data into the actual data. We had done a progressive evaluation scheme to test the performance of our model. Our first evaluation used only data from a single component. Further evaluations use more than one values at the same time. We describe our evaluation method and corresponding results below.

6.2.1 Single Observation Evaluation

We first trained our system only based on a single observed value. We used the data from speed sensor and RPM sensor separately for this. Figure 6.5 represents a part of test data for speed shown graphically. Each of the spikes in it represents the anomalous sudden change in speed caused as a result of the introduction of

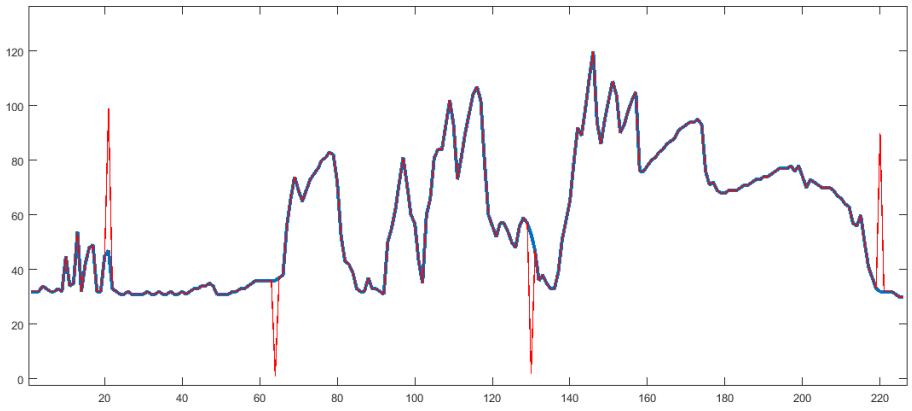


Figure 6.4: Test Data for RPM as a single observation

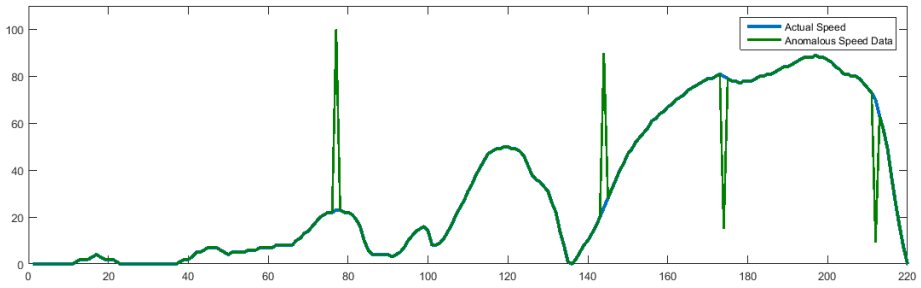


Figure 6.5: Test Data for Speed as a single observation

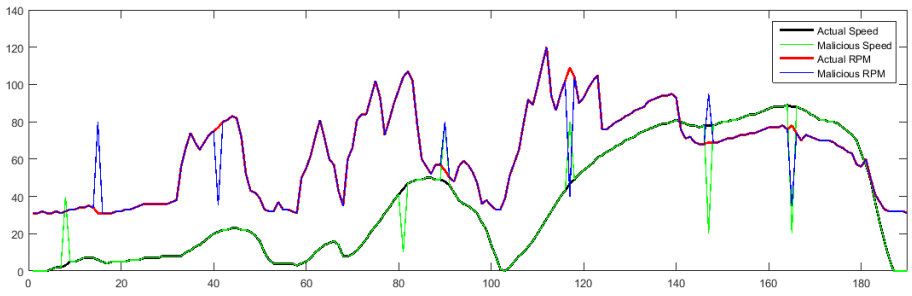


Figure 6.6: Test Data for RPM and Speed together

anomalous data to the real data collected from vehicle. Ideally such a sudden spike is an unsafe state according to our hypothesis. Our generated model was able to detect each of those spikes. In order to make sure that this will work not only for that particular observation, we tried it with RPM sensor data shown in Figure 6.4. In a similar way the spike represents a very sudden change in RPM. We should note that the rate of change in RPM and Speed are different. RPM can increase more rapidly than speed. But since our model is based on real data collected from vehicles, it can detect all those variation which will normally happen in them. The results are concluded in table 6.1. We can see that all different anomalous changes, which cannot correspond to the normal context of a car were detected by our generated model. Moreover at those places which do not have spikes the model did not generate any alert.

No	Type of Change	Speed Alert Status	Result	RPM Alert Status	Result
1	⇑	False	✓	False	✓
2	⇓	False	✓	False	✓
3	⇑⇑	True	✓	True	✓
4	⇓⇓	True	✓	True	✓

Table 6.1: Single Observation Evaluation. ⇑ - Gradual Increase, ⇑⇑ - Sudden Increase, ⇓ - Gradual Decrease ⇓⇓ - Sudden Decrease

6.2.2 Multiple Observations Evaluation

Since our model work well with single observations, we evaluated how it will work while considering multiple observations together in a vector. For this evalu-

ation we chose the speed and RPM observations together as a single vector. Both speed and RPM values are generated at regular intervals and hence we could map every speed value with an RPM value. A part of the anomalous values we generated and tested using our model is represented in figure 6.6. The different spikes represent different anomalous situations which should not happen normally in a vehicle. We tested eight different anomalous situations in it. Each one represents either one of the quantity or both of them being modified which represent a potential malicious state. For example at time 118, the speed suddenly increases while the RPM value decrease, which is an anomalous scenario in a normal running vehicle. Similarly at time around 92 the RPM and speed increase abnormally. After generating the model, we tested these different cases and the evaluation results are described in table 6.2. We can see that each of such situations could be detected by our model and hence the results are promising. But we acknowledge the fact that we need to test our method with more anomalous states of varying degrees.

6.3 Discussion

In this chapter, we propose a Hidden Markov Model based technique to detect anomalous events in a car. As described in the results section, we were able to detect some anomalous states in the sequence of events using such a learned model. However, there is well-documented research which points to scalability issues with Hidden-Markov Models. When the number of states and observations increase, the state transition and emission transition matrices will get bigger and harder to

No	Speed	RPM	Alert Status	Result
1	$\Uparrow\Uparrow$	$\Uparrow\Uparrow$	True	✓
2	$\Uparrow\Uparrow$	$\Downarrow\Downarrow$	True	✓
3	$\Downarrow\Downarrow$	$\Uparrow\Uparrow$	True	✓
4	$\Downarrow\Downarrow$	$\Downarrow\Downarrow$	True	✓
5	$\Uparrow\Uparrow$	$\Longleftrightarrow$	True	✓
6	$\Downarrow\Downarrow$	$\Longleftrightarrow$	True	✓
7	$\Longleftrightarrow$	$\Uparrow\Uparrow$	True	✓
8	$\Longleftrightarrow$	$\Downarrow\Downarrow$	True	✓

Table 6.2: Multiple Observation Evaluation. $\Uparrow\Uparrow$ - Sudden Increase, $\Downarrow\Downarrow$ - Sudden Decrease, $\Longleftrightarrow$ - Normal (from Test data)

manage. Also, the learning will also be badly affected. In this dissertation, however, this research has important implications. Albeit with lesser number of sensors, we showed that the normal sequences can be discerned from the anomalous sequences using a model that is learned from normal operational data of different sensors. This motivate the requirement of an efficient technique which can ingest the normal operational data from a car (and in general cyber-physcial systems), learn their intricate relationships automatically, and use the learned relationships to identify anomalous behaviors. In the later chapters, we work towards such a goal.

Chapter 7: Automatic Behavioral Abstraction Technique for Smart Cyber-Physcial Systems

In this chapter, we develop a novel, scalable, and domain-independent technique, *ABATe* (*A*utomatic *B*ehavioral *A*bstraction to learn operational behaviors specific to a cyber-physical system and use them to identify anomalous behaviors. Attacks on CPS's, often defined as *"intentional actions trying to cause undesired effects in the physical world"* [25], are also perceived deviations from the normal behaviors. Hence, we developed a new technique*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 chapter, we demonstrate *ABATe*'s capabilities using real-world datasets from two different CPS domains. The first dataset(SWaT) [23] 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.

7.1 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 7.1. 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.

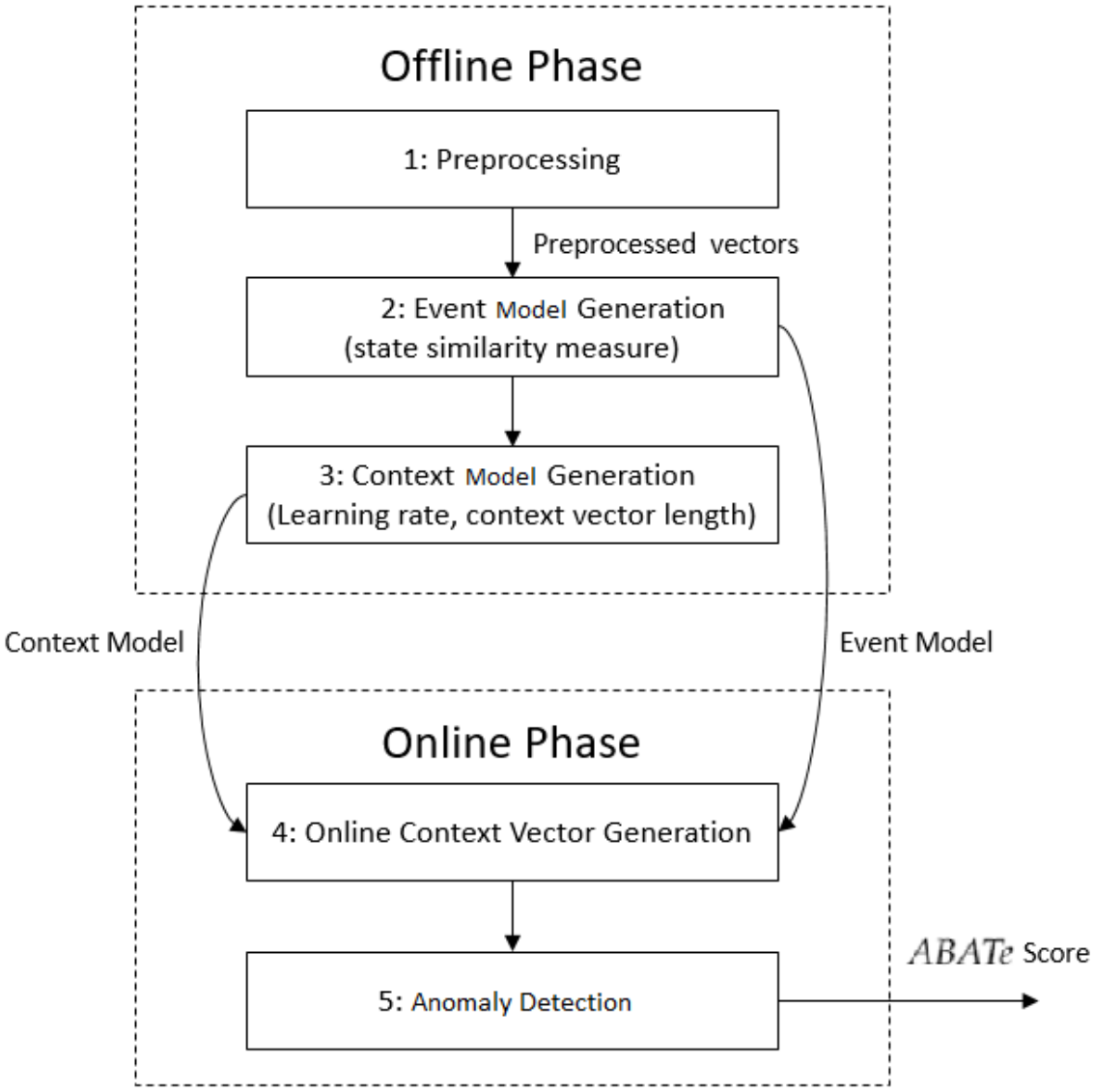


Figure 7.1: *ABATe* Implementation Pipeline

7.1.1 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.

7.1.1.1 State Vector Generation

Let X (Eq:7.1) 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, $r\vec{v}_t$ (Eq:7.2), is defined as a vector of the aggregation of values from all its individual components at that time instant. Let RV_{seq} (Eq: 7.3) be the input sequence of raw vectors. Each $r\vec{v}_t$ 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.

$$X = \{x_1, x_2, ..., x_i, ..., x_n\} \tag{7.1}$$

$$r\vec{v}_t = \langle v_{1,t}, v_{2,t}, ..., v_{i,t}, ..., v_{n,t} \rangle \tag{7.2}$$

$$RV_{seq} = r\vec{v}_1, r\vec{v}_2, ..., r\vec{v}_i, ..., r\vec{v}_t$$
$$where, \tag{7.3}$$
$$r\vec{v}_t \ : \ raw\ vector\ at\ time\ t$$

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's have constrained behavior. Hence, we add only those $r\vec{v}_t$'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 $r\vec{v}_t$'s and fold them as the same state, if the distance falls below an empirically found state transition threshold, τ_{sim}. Each state in S now corresponds to a legal state in the respective CPS.

We label each state $\vec{s_i} \in S$ randomly and generate hot vectors for each of them. The generated hot vector set HS has a $\vec{hv_i}$ corresponding to each $\vec{s_i} \in S$. We generate the hot vectors to make sure that each $\vec{hv_i}$ 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.

7.1.1.2 Context Vector Generation

In this task, *ABATe* learns context vectors that abstract the perceived normal behavior of the CPS for each state $\vec{hv_i} \in HS$. To learn context vectors, *ABATe* finds an embedding for each state $\vec{hv_t}$ by capturing their distributional behavior in the normal operational data. In mathematics, an embedding is defined as *"a*

representation of a topological object, manifold, graph, field, etc. in a certain space in such a way that its connectivity or algebraic properties are preserved"[1]. Neural networks are successfully used to generate such embeddings in domains like Natural Language Processing(NLP) [48] that learns the distributional behavior of words in natural language text. In a similar way, *ABATe* learns the embeddings for each state vector, $\vec{hv_i} \in HS$ using neural networks and CPS's normal data. These embeddings encapsulate the distributional behavior and encode the contextual information with them. For example, let $\vec{c_i}, \vec{c_j}$ be the learned embeddings corresponding to $\vec{hv_i}, \vec{hv_j}$. Then, the vector similarity between $\vec{c_i}$ & $\vec{c_j}$ will be high if $\vec{hv_i}$ & $\vec{hv_j}$ 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 $\vec{hv_i}$ into $\vec{c_i}$ as presented in Eq: 7.4 and Eq: 7.5. The network has an input layer with size equal to the number of states in HS ($HS_{size} =\mid HS \mid$). 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 layer with input size as the context vector length (C_{size}) and output size as HS_{size}. Once training is done, the first layer will act as a lookup function and convert the state vector $\vec{s_i}$ to context vector $\vec{c_i}$. The $\vec{c_i}$ is a condensed representation of $\vec{hv_i}$ with encoded context or their distribution in the

[1]http://mathworld.wolfram.com/Embedding.html

normal data. In addition, it also learns implicit relationships between states based on their contexts.

Input Layer:

$$
\begin{aligned}
&\vec{c_i} = f(\vec{hv_i}) \\
&f(x) = Wx + B \\
&\quad where, \\
&\quad\quad \vec{hv_i}\text{: Hot Vector corresponding to } \vec{s_i} \\
&\quad\quad W\text{: Matrix with size } HS_{size} \times C_{size} \\
&\quad\quad B\text{: vector with size } C_{size} \\
&\quad\quad \vec{c_i}\text{: Context Vector corresponding to } \vec{s_i}
\end{aligned}
\tag{7.4}
$$

Soft-max Layer:

$$
\begin{aligned}
&Output = SoftMax(\vec{c_i}) \\
&\quad where, \\
&\quad\quad \vec{c_i}\text{: Context Vector corresponding to } \vec{s_i}
\end{aligned}
\tag{7.5}
$$

To train the network, we introduce a new variable $\vec{chv_i}$ corresponding to each $\vec{hv_i}$. Each $\vec{chv_i}$ is a hot vector and assumes no relationship between any existing $\vec{hv_i}$ 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 $\vec{hv}_1, \vec{hv}_2, ..., \vec{hv}_i, \vec{hv}_{i+1}, ...$ be a sequence which appears in the normal operation of the CPS. Now for each $\vec{hv}_i$, we generate tuples $\langle \vec{hv}_{i-k}, \vec{chv}_i \rangle \forall_{k=0}^{k=cw\_size}$. 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 $\vec{hv}_{i-k} \forall_{k=0}^{k=cw\_size}$. 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.

7.1.2 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 the live inputs from different components in the system and generates an $ABATe_{score}$ that help 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.

Algorithm 1 Off-line Learning Phase

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

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

The first step in this phase is the aggregation and generation of raw vectors, $InputRV_{seq} = \vec{rv}'_1, \vec{rv}'_2, \vec{rv}'_3, ..., \vec{rv}'_t, ...$ using inputs from different components in the CPS, where $\vec{rv}'_t$ is the input vector at time t. As described in section 7.1.1.1, each $\vec{rv}'_t$ will be a point the same n-dimensional space. We choose a state $\vec{s}'_i \in S$ corresponding to each $\vec{rv}'_t$ such that the Euclidean distance between $\vec{s}_i$ and $\vec{rv}'_t$ is minimum. Now a sequence of states $InputState_{seq} = \vec{s}'_1, \vec{s}'_2, ...\vec{s}'_t, ...$ will be generated where $\vec{s}'_t$ is the chosen state corresponding to each $\vec{rv}'_t$. Let EU_t be the Euclidean distance between a raw vector $\vec{rv}'_t$ and $\vec{rv}_i$ corresponding to $\vec{s}'_i$. 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 7.1.1.2, the context vectors encapsulate the context of each vector and hence they are used to detect such anomalies. Let $InputCV_{seq} = \vec{c}_1, \vec{c}_2, ...\vec{c}_t, ...$ be the context vector sequence corresponding to $InputState_{seq}$, where $\vec{c}_t$ is the context vector corresponding to state $\vec{s}_t$. In *ABATe*, we use the cosine distance (Eqn: 7.6) of the current context vector $\vec{c}_t$ and the previous context vector $\vec{c}_{t-1}$ 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 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: 7.7. The first component in the $ABATe_{score}$ corresponds to the dissimilarity of current raw vector $r\vec{v}'_t$ 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 low 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 $\tau_{anomaly}$. The operations of this stage are described in algorithm 2.

$$
\begin{aligned}
ContextSim_t &= 1 - \frac{\vec{c}'_t \cdot \vec{c}'_{t-1}}{\|\vec{c}'_t\| \|\vec{c}'_{t-1}\|} \\
&\textit{where,} \\
&\vec{c}'_t, \vec{c}'_{t-1}\textit{: Context Vectors corresponding to } \vec{s}'_t \textit{ and } \vec{s}'_{t-1}
\end{aligned}
\tag{7.6}
$$

$$
\begin{aligned}
ABATe_{score} &= e^{(EU_t)} * ContextSim_t \\
&\textit{where,} \\
&ContextSim : \textit{ Contextual similarity from Eqn: 7.6} \\
&EU_t : \textit{ Euclidean distance of } \vec{rv}'_t \textit{ to } \vec{s}'_t
\end{aligned}
\tag{7.7}
$$

Algorithm 2 On-line Monitoring Phase

Input:
$InputRV_{seq} \leftarrow \vec{rv'_1}, \vec{rv'_2}, ..., \vec{rv'_i}, ...$
StateMdl: Raw vector to State vector mapping
CtxMdl: State vector to Context vector mapping
Output:
$ABATe_{Score}$

1: **procedure** MONITORINGPHASE($InputRV$)
2: **for each** $rv'_i \in InputRV_{seq}$ **do**
3: $\vec{s'_t}, EU_t \leftarrow getSim(StateMdl, \vec{rs'_t})$
4: $ContextSim_t \leftarrow$*Eqn 7.6*$(CtxMdl, \vec{s'_t}, \vec{s'_{t-1}})$
5: $ABATe_{score} \leftarrow$Eqn 7.7$(n, EU_t, ContextSim_t)$
6: **if** $ABATE_{Score} > \tau_{anomaly}$ **then**
7: *report Anomaly*
8: **else**
9: *report Normal*
10: **end if**
11: **end for**
12: **end procedure**

7.2 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 7.1. 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, τ_{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 7.1.1.2. 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 $r\vec{v}'_t$. Then a state label $\vec{s}'_t \in S$ is determined from the event model by choosing the state having the closest Euclidean distance from $r\vec{v}'_t$. 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.

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: 7.6 and it is then used in Eqn: 7.7 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, window-quartile-25, percent change, and so forth on the sequence of $ABATe_{score}$'s. Their evaluation can be found in section 8. 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 es-

timated 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.

Chapter 8: 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 chapter, 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 [23] (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.

8.1 SWaT dataset Evaluation

The Sewage Water Treatment (SWaT) dataset from Goh et al. [23] 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. “Pretreatment” stage
3. “Ultrafiltration and backwash” stage
4. “DeChlorination” 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’s 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 only use the measurements from its components.

8.1.1 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 mentioned in section 7.2, 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.

The first step in *ABATe* is to generate a perceived normal behavioral model. We used normal data from the first 7 days to train this model (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 $ABATe_{scores}$ were generated as described in section 7.2. Different techniques applied to the generated sequence of $ABATe_{scores}$ to detect the presence of different attacks include *gaussian-average*, *window-median*, *window 25 quantile*, *window 75 quantile*, and *percentage change*.

8.1.2 Results

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 the y-axis against specificity (False positive

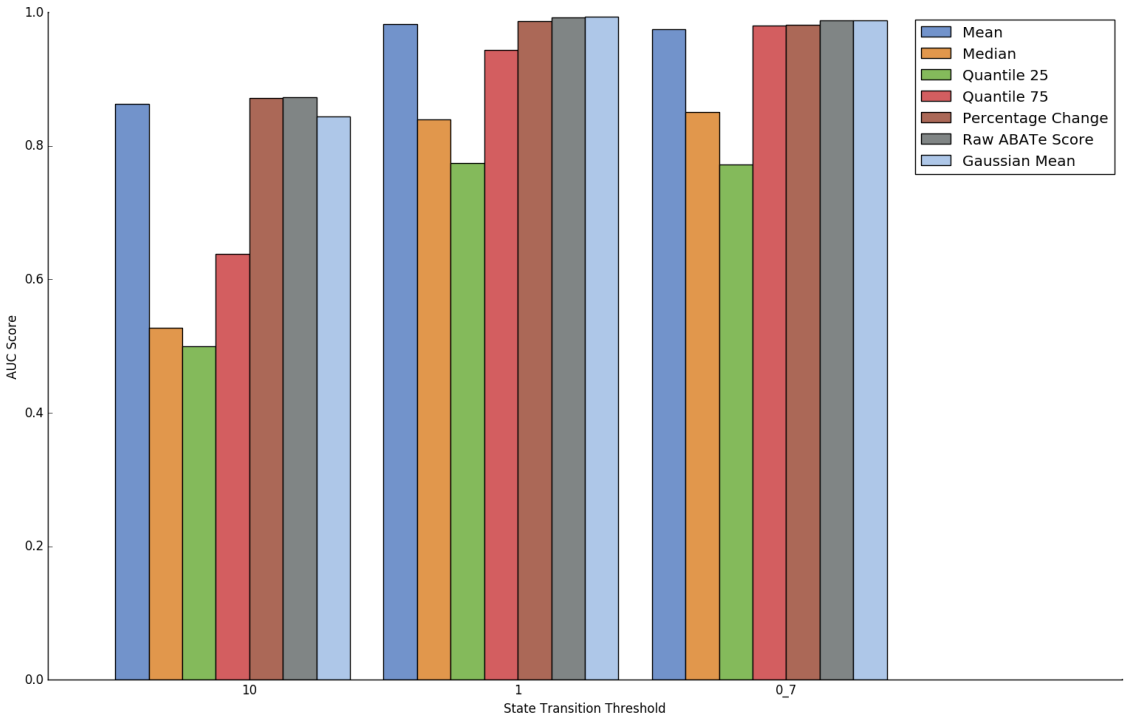


Figure 8.1: *ABATe* SWaT data AUC Plots

rate) on the x-axis. 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.

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, 25 quantile, 75 quantile, and gaussian mean along with raw $ABATe_{score}$'s. One other measurement, 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 8.1 depicts the clustered bar graph of AUROC values for these different metrics. On the

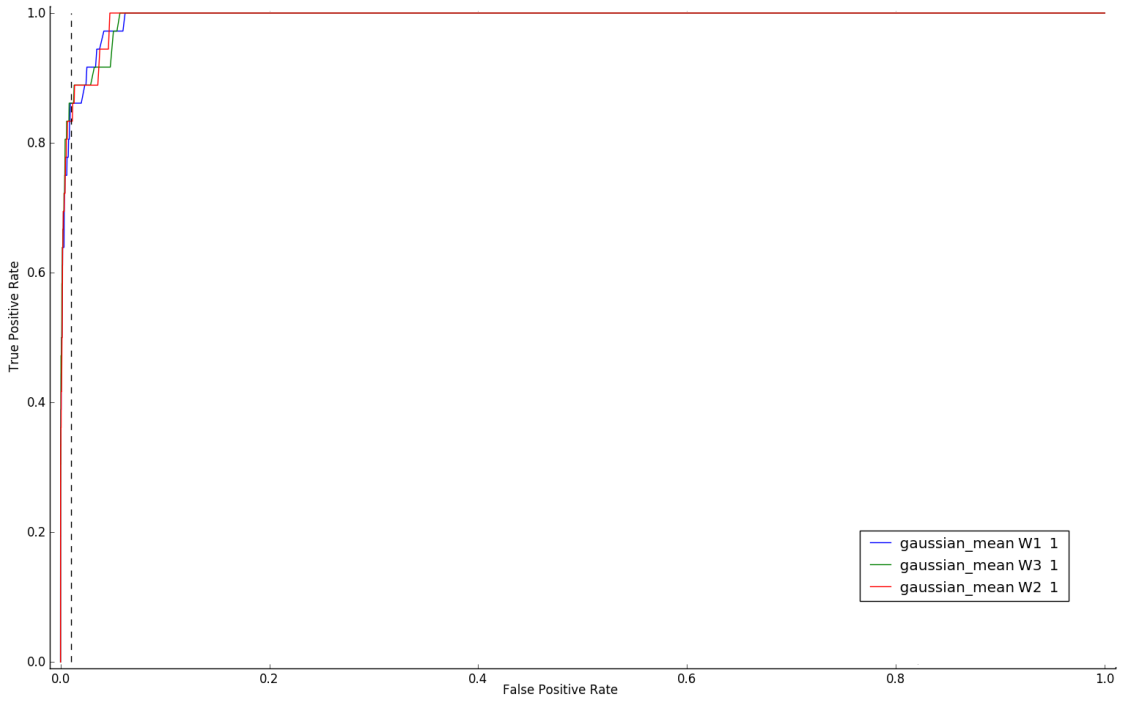


Figure 8.2: *ABATe* SWaT data ROC Plots: Window Comparison using Gaussian Mean Window

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 the stream of $ABATe_{score}$ values. The window size used for the window-based techniques were fixed at 10. From the graphs, we can see that moving window based measurements using mean metrics performed significantly better than quantile based moving window approaches. However, it can also be seen that non-window approaches like percent change based method and raw $ABATe_{score}$ based techniques performed close to window based mean metric approaches. Figure 8.4 plots the ROC curves for the most promising techniques, gaussian mean, raw $ABATe_{score}$, and percent change. From this plot, we see that the gaussian mean technique performs slightly better than others where 88.8% true positive rate is observed at 1.3% false positive rates.

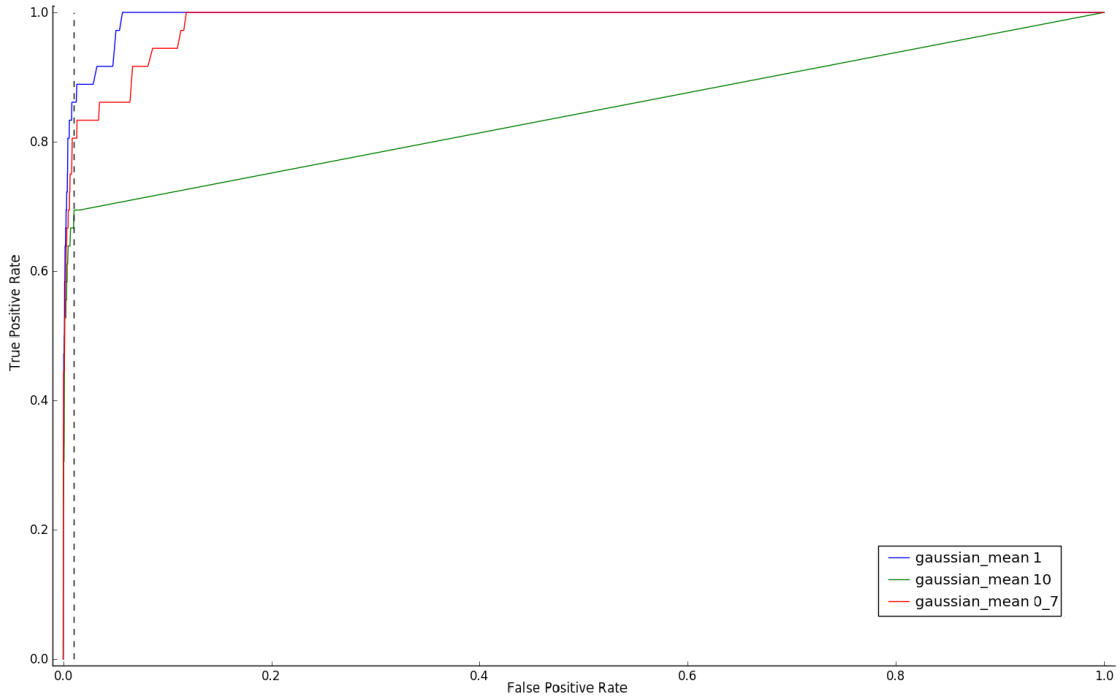


Figure 8.3: *ABATe* SWaT data ROC Plots: State Transition Threshold Comparison at Window size = 3

We detect an attack when there is enough change in the $ABATe_{score}$ during the annotated attack regions in the SWaT dataset while a single anomalous variation during a non-attack period is considered as a false positive. As a result, the number of false positives and true negatives are very high considering the true positives and false negatives.The confusion matrix in the former case will have 32 true positives and 6 false negatives. However, a 1.3% false positive corresponds to more than 6k out of the total 500k true negatives. Just using raw $ABATe_{score}$ can achieve only 83% true positive rate at 1.5% false positive rates.

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 8.3 shows that maximum performance is achieved at 1% state transition

threshold. Figure 8.3 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, 88.8% true positive rate is observed at 1.3% false positive rate for a state threshold of 1.0 while at 0.7%, the true positive rate reduces to 82.9% for a similar false positive rate. 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 8.2 plots the performance of *ABATe* with context windows 1, 2, and 3. In this case, we can see that top performance is delivered by the context window 3 although all the three showed comparable performance.

8.2 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's context from various com-

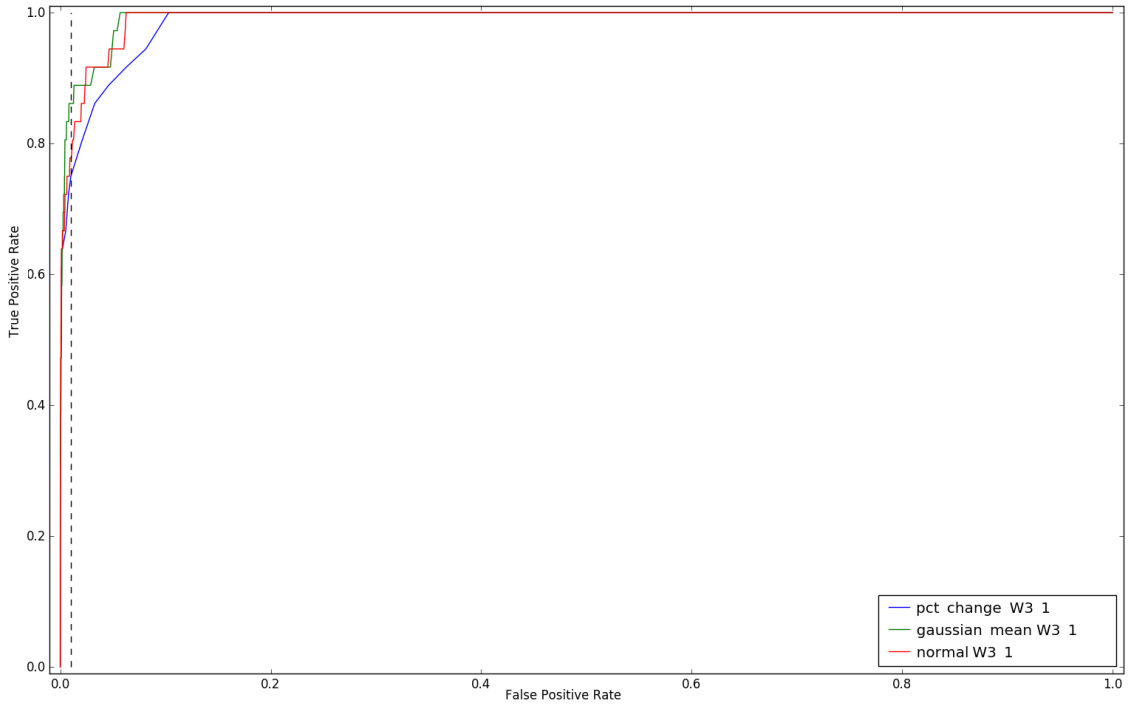


Figure 8.4: *ABATe* SWaT data ROC Plots: Promising Technique ROC Curve Comparison

ponent 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*.

8.2.1 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, $r\vec{v}_t$. 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.

8.2.2 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} x 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 maximum $ContextSim$ score) is described in table 8.1. 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.

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

Component	s_i	s_j
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
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

Table 8.1: Comparing $s_i \in S_{high\_speed}$ and $s_j \in S_{high\_speed}$ with large $ABATe_{score}$

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.

8.2.3 Simulated Attack Detection

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 about 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).

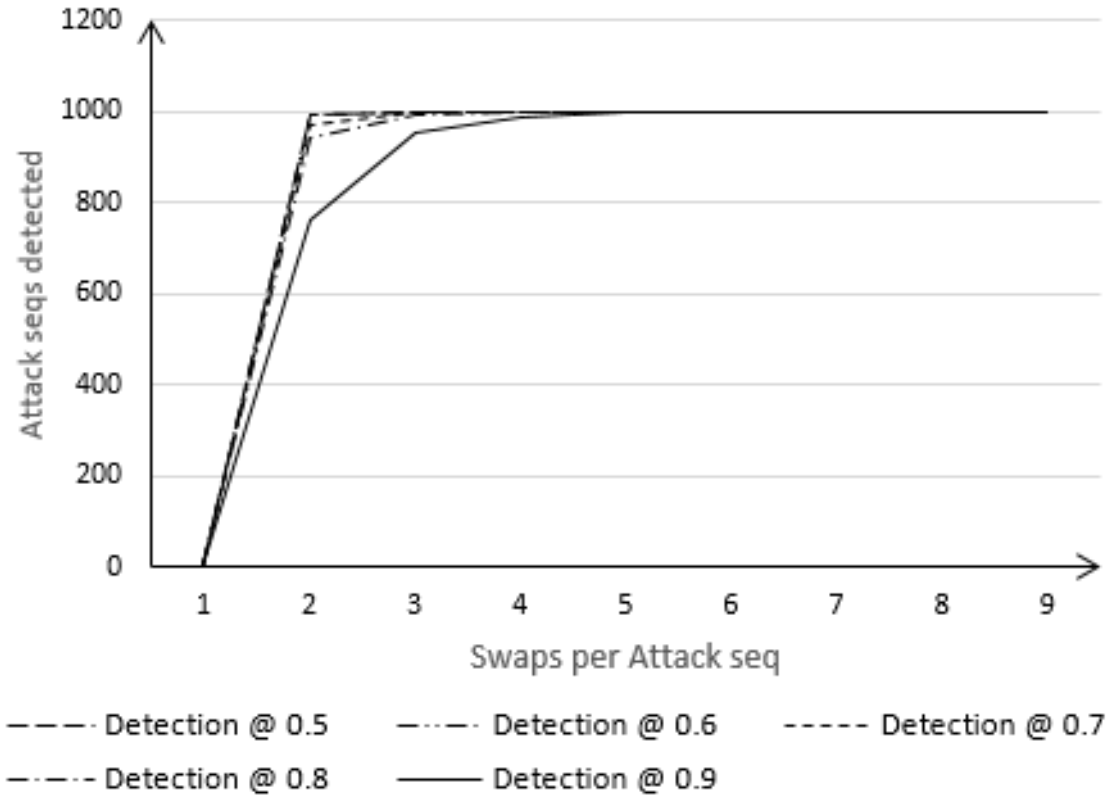


Figure 8.5: *ABATe* Injection Detection Performance

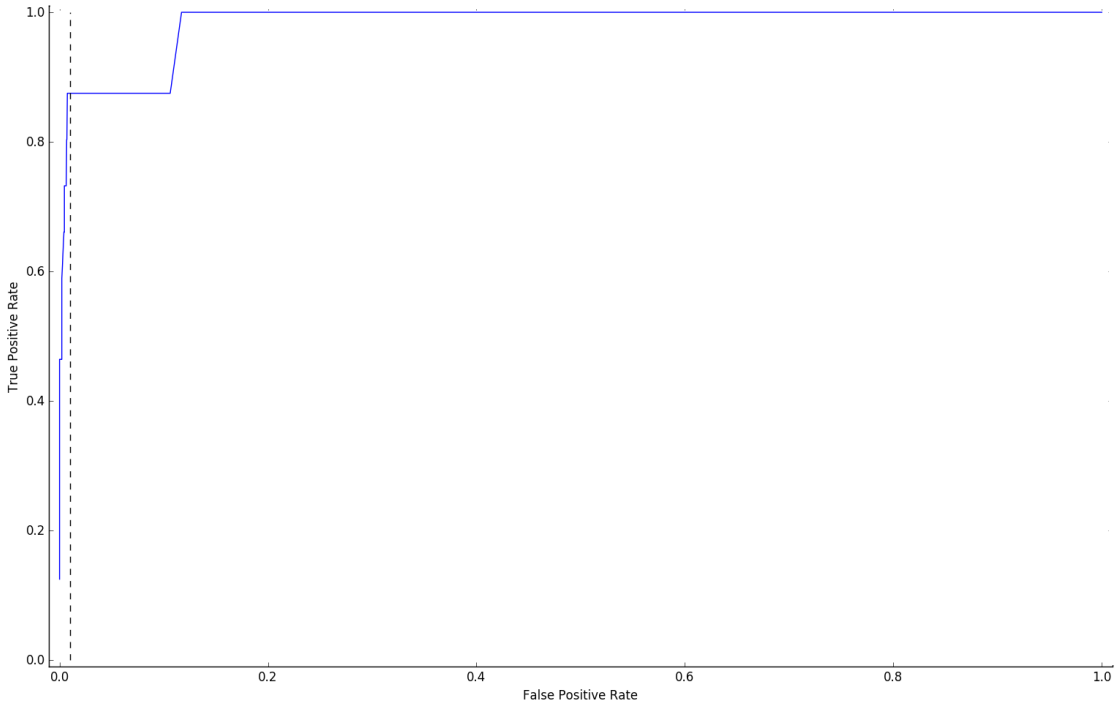


Figure 8.6: *ABATe* Car data ROC curve

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 8.6 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 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 gener-

ated 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 8.5 when the $\tau_{anomaly}$ is varied. Each line represents a specific value for $\tau_{anomaly}$. 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.

8.3 Synthetic Dataset Evaluation

Sections 8.1 and 8.2 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*.

8.3.0.1 Synthetic Dataset Generation

HMM's (Hidden Markov Models) have been used to generate state sequences with fixed probabilities [13]. 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 8.7. 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 three different probabilities for this network. Δ_{ss} representing the probability of each

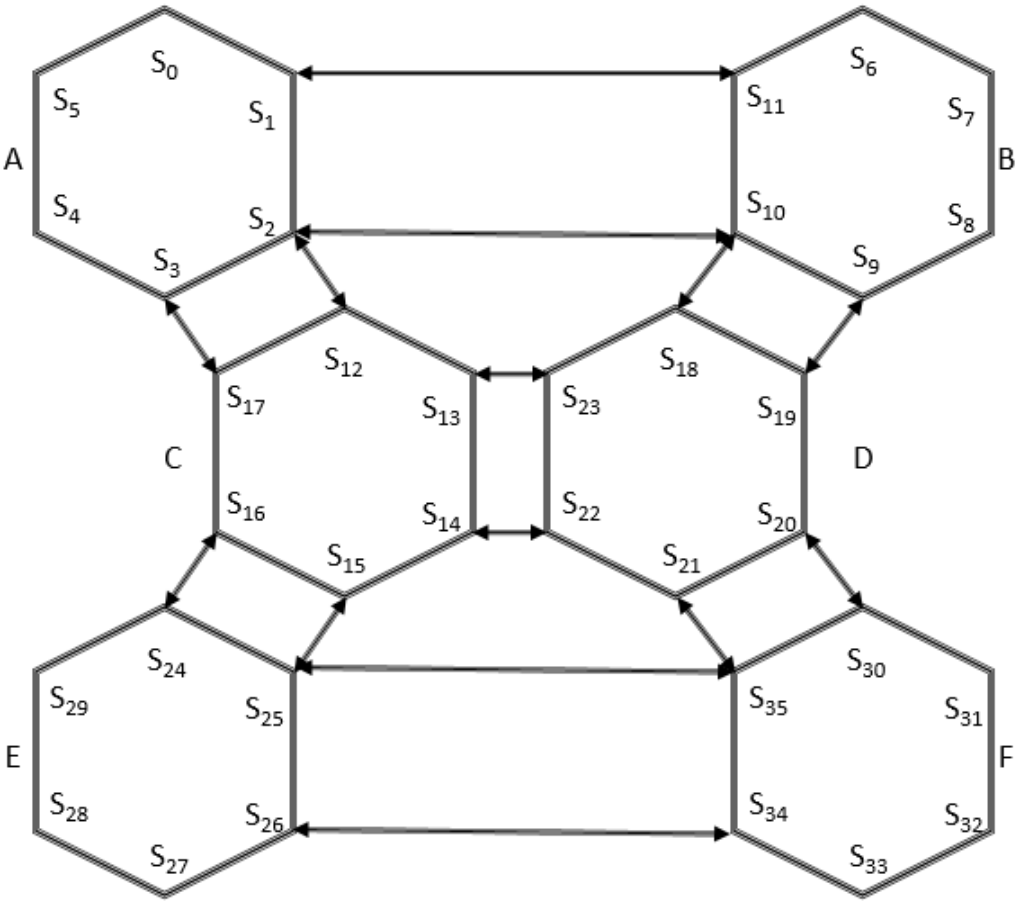


Figure 8.7: Synthetic Data Generation: State Diagram

state to itself, Δ_{sa} representing transition probability of moving to another state in the same hexagon and Δ_{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, $S_0 \rightarrow S_1, S_6 \rightarrow S_7, S_{30} \rightarrow S_{31}$ etc. have the same probability Δ_{sa}, and $S_1 \rightarrow S_{11}, S_{14} \rightarrow S_{22}$ etc. have the transition probability Δ_{ia}. By tweaking these three variables we generate different sequences. If there is no edge connecting two vertices, they have zero probabilities. For instance $S_1 \rightarrow S_3, S_4 \rightarrow S_{29}$, 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 $\Delta_{ia} = \Delta_{sa} = \Delta_{ss}$, sequence with Δ_{ia} greater than Δ_{sa} and Δ_{ss}, and sequence with Δ_{sa} greater than Δ_{sa} and Δ_{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 $s0 \rightarrow s6$ and regenerating the sequence will produce a sequence with $s0 \rightarrow s6$ 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).

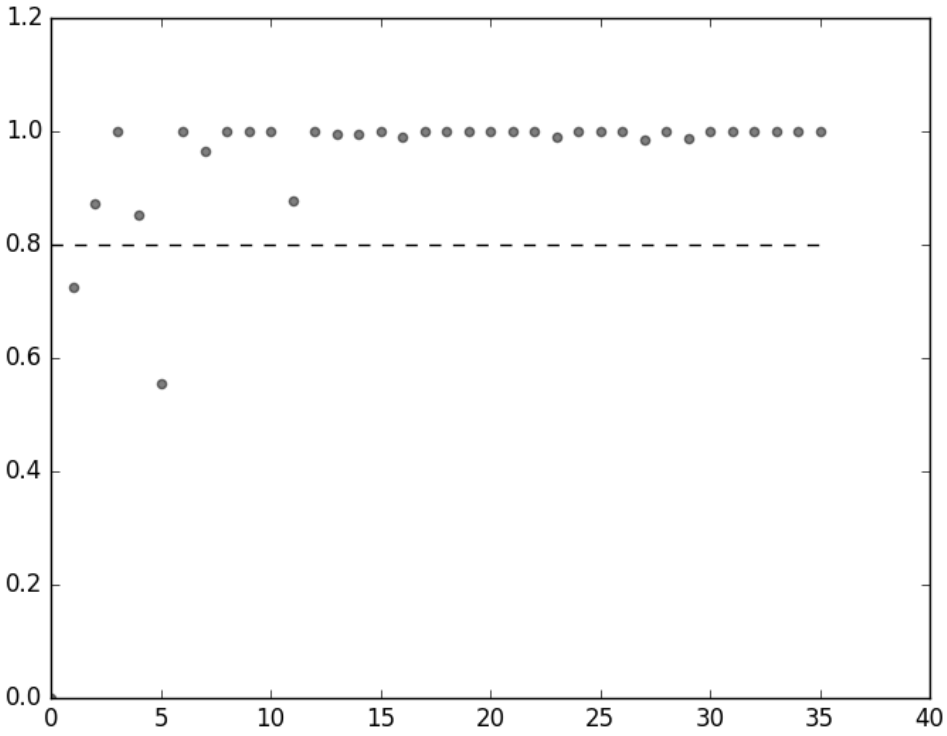


Figure 8.8: State 0 Anomaly Scores with Context 1

8.3.1 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: 7.7 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.

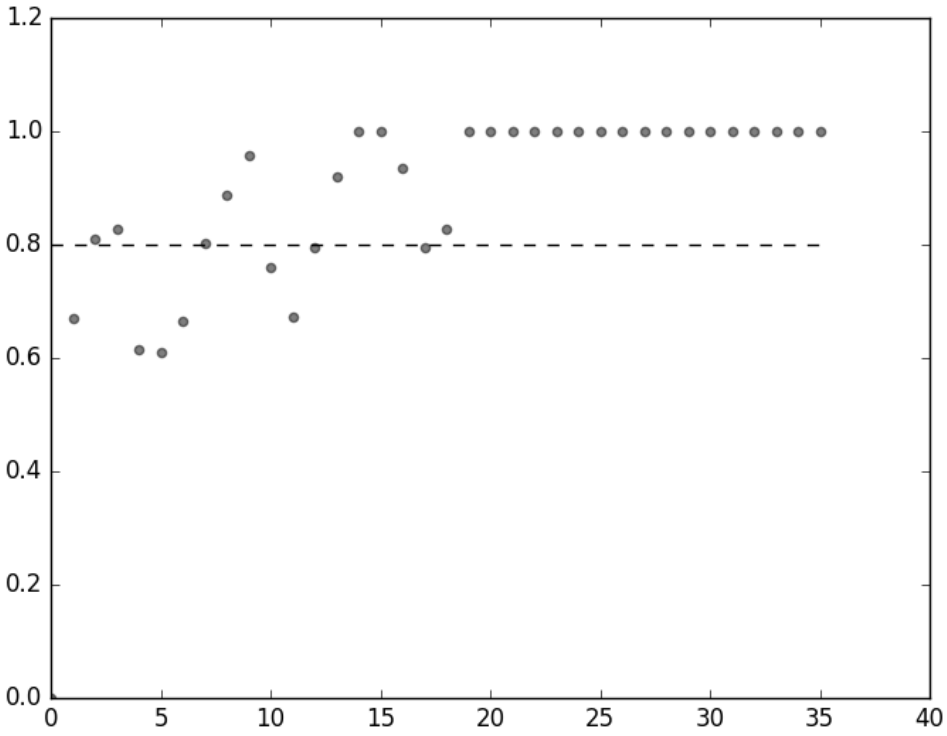


Figure 8.9: State 0 Anomaly Scores with Context 4

Figures 8.8 and 8.9 present the *ContextSim* scores for the state s_0 with context windows sizes 0 and 4. From the state diagram in Figure 8.7, 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_0 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 8.2. 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 $s_0 \rightarrow s_9$ 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.

End State	Path
s_1	$s_0 \rightarrow s_1$
s_4	$s_0 \rightarrow s_5 \rightarrow s_4$
s_5	$s_0 \rightarrow s_5$
s_6	$s_0 \rightarrow s_1 \rightarrow s_{11} \rightarrow S_6$
s_{10}	$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_{10}$
s_{11}	$s_0 \rightarrow s_1 \rightarrow s_{11}$

Table 8.2: Synthetic Dataset State transition path for State s_0

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 8.2 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 also without much increasing the FPR's.

8.4 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 7.1, the first task is the generation of raw vector, $r\vec{v}_i'$ which is a straight forward function with complexity of $O(1)$. The next task is to choose a $\vec{hv}_i'$ 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.

Chapter 9: Conclusion & Future Directions

Rapid advancements in control systems, sensing, networking, and artificial intelligence resulted in the large-scale deployment of smart Cyber-Physical Systems in a variety of fields including medicine, manufacturing, automatic pilot avionics, and so forth. They control many critical infrastructure in their respective domains and provide better visibility, control, efficiency, reliability, and safety. However, these advanced capabilities made them a lucrative target for hackers. The affected users ranged from normal residents of a city when the hackers took down a power grid in Ukraine to national governments when Stuxnet malware infected a nuclear power plant's programmable logic circuits (PLC) for physically damaging them. Apart from such intentional attacks against cyber-physical systems, recent incidents involving aircraft disasters caused by device malfunctioning and human errors also put their safety and reliability into question. In this dissertation, we develop novel techniques to detect abnormal behaviors in smart cyber-physical system using the data emanating from them.

First, we hypothesized that the intricate relationships between different components of a smart cyber-physical system are well-represented in the data emanating from them. We developed a knowledge-graph based solution that infers complex

contexts using simpler ones defined using semantic technologies. Such a system can detect several abnormal scenarios. Unlike simple rule-based systems, it could infer complex contexts from simpler contexts defined by the domain experts. However, learning newer and unintuitive rules is difficult and the accuracy in attack detection is directly dependent on the closure of all the rules. Further, from our experimentation with the Hidden-Markov models on the automotive domain, we showed that complex relationships between the components of a cyber-physical system could be used to detect abnormal behaviors.

In another significant contribution in this dissertation, we developed a scalable and domain independent solution to learn typical behaviors in smart cyber-physical systems. For achieving this goal, we developed a neural network based solution that generates a latent space to abstract typical domain behaviors using the plethora of data available from these smart systems. Our evaluations using an automotive dataset demonstrated the context abstraction capabilities of our technique. Further, we developed a technique which used the generated latent space to detect abnormal scenarios in those cyber-physical systems. We demonstrated our technique's capability to detect attacks using two real-world datasets; SWaT dataset and an automotive dataset. The SWaT dataset consists of operational data from a scaled down water treatment plant with 51 sensors from 7 different phases. Using our technique, we were able to detect 88% of attack scenarios with a low false positive rate of around 1%. In the automotive dataset, we detected some specifically hand-crafted attack scenarios and false data injection based attack scenarios.

Evaluations using real-world datasets are important in cyber-physical systems.

However, we have limited datasets available in the public domain. A key engineering contribution in this dissertation is the development of a new automotive dataset. In the process, we evaluated several data collection techniques from cars like ELM 327 chipset based techniques, STN1100 chipset based techniques, Arduino with CAN-Bus shield, OpenXC, and so forth. Eventually, we used OpenXC based data collection because of the rich API support from its parent company Ford. Our new automotive dataset comprised close to 1000 miles or 25 hours of real-driving data from 11 different sensors in a modern car. We ensured that the collected data included diverse driving conditions from long and short drives on highways, countrysides, mountainous terrains, and rainy conditions. We believe that this dataset can promote meaningful research in this domain in a similar way it helped our research.

9.1 Future Directions

In this dissertation, we showed that a generic solution can be employed to learn the typical behaviors of a smart cyber-physical system and such behaviors can be used to detect anomalous behaviors. Some future extensions to our work are discussed in this section.

The first logical step after anomaly detection is the invocation of mitigation strategies or preventive measures. A possible extension of our work is to develop techniques which differentiate contexts abstracted by *ABATe* using the context vectors. Such abstracted contexts can be linked to a policy framework to invoke mitigation strategies. A simple example from the automotive domain is when the current

context corresponds to driving in a high-way policies like "alerting the user" would be the best choice while when the context corresponds to a stationary vehicle, a policy can suggest "turn off the ignition". The second research direction can be the identification of anomaly cause. Unlike some data-oriented techniques for anomaly detection, our learned model abstracts the context of events happening in the smart cyber-physical system. The $ABATe_{score}$, and the events used for calculating it, indicates which events might have caused the anomaly. Instead of learning the events and their relationships learning states per component in the smart cyber-physical system and learning their combined relationship behavior is worth investigation. However, we might encounter big-data related issues while learning such a model. For example, with SWaT dataset, if we create a dataset for learning such a combined model, the combined training set may contain 1 billion to 6 billion training instances depending on the window size we choose. We should explore big-data solutions to learn in such scenarios.

Another research direction is the reduction in false positive rates. It is sometimes critical because of the huge cost associated with restarting large scale systems. In our research, we considered that little information is available about the behaviors of the smart cyber-physical system and learn automatically from the data. However, we can employ a properly designed knowledge-graph in conjunction with our approach to improve its efficiency. For example, in the state generation phase, we give equal weightage to all the components of the cyber-physical system when in many real scenarios, some entities are not at all related. Manual design of such information will reduce the usability of our technique in different scenarios. However,

a well-populated knowledge graph shipped with the technique can carry such information and can help improve our system. Exploring such fusion of knowledge-graph based techniques to our system might provide interesting results.

Bibliography

[1] Sridhar Adepu and Aditya Mathur. Distributed attack detection in a water treatment plant: Method and case study. *IEEE Transactions on Dependable and Secure Computing*, 2018.

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

[3] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, EMANUELE DELLA VALLE, and Michael Grossniklaus. C-sparql: a continuous query language for rdf data streams. *International Journal of Semantic Computing*, 4(01):3–25, 2010.

[4] Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. Querying rdf streams with c-sparql. *ACM SIGMOD Record*, 39(1):20–26, 2010.

[5] 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., 2017.

[6] Andre Bolles, Marco Grawunder, and Jonas Jacobi. Streaming sparql-extending sparql to process data streams. In *European Semantic Web Conference*, pages 448–462. Springer, 2008.

[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*, 2014.

[8] Suratna Budalakoti, Ashok N Srivastava, Ram Akella, and Eugene Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. *Nasa Technical Report Server*, 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] Defense Use Case. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

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

[12] Varun Chandola. *Anomaly detection for symbolic sequences and time series data*. PhD thesis, University of Minnesota, 2009.

[13] 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.

[14] Hong Chen. Applications of cyber-physical system: a literature review. *Journal of Industrial Integration and Management*, 2(03):1750012, 2017.

[15] 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.

[16] Louis Columbus. Where iot can deliver the most value in 2018, 2018.

[17] 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.

[18] Prajit Kumar Das, Sandeep Narayanan, Nitin Kumar Sharma, Anupam Joshi, Karuna Joshi, and Tim Finin. Context-sensitive policy based security in internet of things. In *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[19] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[20] Delia Ioana Dogaru and Ioan Dumitrache. Cyber-physical systems in healthcare networks. In *2015 E-Health and Bioengineering Conference (EHB)*, pages 1–4. IEEE, 2015.

[21] Christopher E Everett and Damon McCoy. Octane (open car testbed and network experiments): Bringing cyber-physical security research to researchers and students. In *CSET*, 2013.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] Bernardo Cuenca Grau, Christian Halaschek-Wiener, and Yevgeny Kazakov. History matters: Incremental ontology reasoning using modules. In *The Semantic Web*, pages 183–196. Springer, 2007.

[28] Bogdan Groza, Stefan Murvay, Anthony Van Herrewege, and Ingrid Verbauwhede. Libra-can: Lightweight broadcast authentication for controller area networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(3):90, 2017.

[29] Ahmed Hazem and Hossam AH Fahmy. Lcap-a lightweight can authentication protocol for securing in-vehicle networks. In *10th escar Embedded Security in Cars Conference, Berlin, Germany*, volume 6, 2012.

[30] 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.

[31] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. Security threats to automotive can networks–practical examples and selected short-term countermeasures. In *Computer Safety, Reliability, and Security*, pages 235–248. Springer, 2008.

[32] Zhichuan Huang, David Corrigan, Sandeep Narayanan, Ting Zhu, Elizabeth Bentley, and Michael Medley. Distributed and dynamic spectrum management

in airborne networks. In *Military Communications Conference, MILCOM 2015-2015 IEEE*, pages 786–791. IEEE, 2015.

[33] 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.

[34] Min-Joo Kang and Je-Won Kang. Intrusion detection system using deep neural network for in-vehicle network security. *PloS one*, 11(6):e0155781, 2016.

[35] Min-Ju Kang and Je-Won Kang. A novel intrusion detection method using deep neural network for in-vehicle network security. In *Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd*, pages 1–5. IEEE, 2016.

[36] Vishal Maruti Karande, Sandeep Nair Narayanan, Alwyn Roshan Pais, and N Balakrishnan. Testing resilience of router against denial of service attacks. In *Trends in Network and Communications*, pages 107–116. Springer, 2011.

[37] 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.

[38] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 447–462. IEEE, 2010.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] Pavel Laskov, Konrad Rieck, Christin Schäfer, and Klaus-Robert Müller. Visualization of anomaly detection using prediction sensitivity. In *Sicherheit*, volume 2, pages 197–208, 2005.

[44] Yining Li, Jing Wu, and Shaoyuan Li. State estimation for distributed cyber-physical power systems under data attacks. *International Journal of Modelling, Identification and Control*, 26(4):317–323, 2016.

[45] 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.

[46] 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.

[47] Farhad Mehdipour. Smart field monitoring: An application of cyber-physical systems in agriculture (work in progress). In *2014 IIAI 3rd International Conference on Advanced Applied Informatics*, pages 181–184. IEEE, 2014.

[48] 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.

[49] Charlie Miller and Chris Valasek. Adventures in automotive networks and control units. *DEF CON*, 21:260–264, 2013.

[50] Charlie Miller and Chris Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015:91, 2015.

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

[52] 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.

[53] Michael Müter and Naim Asaj. Entropy-based anomaly detection for in-vehicle networks. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 1110–1115. IEEE, 2011.

[54] Narayanan Sandeep Nair, Joshi Anupam, and Bose Ranjan. Abate: A deep network based approach to detect attacks in cyber-physical systems. In *Submitted (Under review)*, 2018.

[55] Narayanan Sandeep Nair, Ashwin Ganeshan, Karuna Joshi, Tim Oates, Anupam Joshi, and Finin Tim. Early detection of cybersecurity threats using collaborative cognition. In *4th IEEE International Conference on Collaboration and Internet Computing*, 2018.

[56] Sandeep Nair, Sudip Mittal, and Anupam Joshi. Using semantic technologies to mine vehicular context for security. In *37th IEEE Sarnoff Symposium (2016)*, 2016.

[57] 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.

[58] Sandeep Nair Narayanan, Kush Khanna, Bijaya Ketan Panigrahi, and Anupam Joshi. Security in smart cyber-physical systems: A case study on smart grids and smart cars. In *Smart Cities Cybersecurity and Privacy*, pages 147–163. Elsevier, 2019.

[59] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi. Obd_securealert: An anomaly detection system for vehicles. In *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[60] Sandeep Nair Narayanan, Alwyn Roshan Pais, and Radhesh Mohandas. Detection and prevention of sql injection attacks using semantic equivalence. In *Computer Networks and Intelligent Computing*, pages 103–112. Springer, 2011.

[61] 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.

[62] Timothy J O'Shea, T Charles Clancy, and Robert W McGwier. Recurrent neural radio anomaly detection. *arXiv preprint arXiv:1611.00301*, 2016.

[63] 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.

[64] Jeff Z Pan, Edward Thomas, Yuan Ren, and Stuart Taylor. Exploiting tractable fuzzy and crisp reasoning in ontology applications. *IEEE Computational Intelligence Magazine*, 7(2):45–53, 2012.

[65] Steven W Popper, Steven C Bankes, Robert Callaway, and Daniel DeLaurentis. System of systems symposium: Report on a summer conversation. *Potomac Institute for Policy Studies, Arlington, VA*, 320, 2004.

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

[67] Vishal Rathod, Sandeep Narayanan, Sudip Mittal, and Anupam Joshi. Semantically rich, context aware access control for openstack. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 460–465. IEEE, 2018.

[68] Arya Renjan, Karuna Joshi, Sandeep Nair Narayanan, and Anupam Joshi. Dabr: Dynamic attribute-based reputation scoring for malicious ip address detection. In *IEEE Intelligence and Security Informatics (ISI)*, 2018.

[69] Narayanan Sandeep Nair Renjan, Arya and Joshi Karuna. A policy based framework for privacy-respecting deep packet inspection of high velocity network traffic. In *2019 IEEE 5th IEEE International Conference on Big Data Security on Cloud (BigDataSecurity 2019)*. IEEE, 2019.

[70] Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, et al. Fully convolutional neural network for fast anomaly detection in crowded scenes. *arXiv preprint arXiv:1609.00866*, 2016.

[71] Borja Bordel Sánchez, Ramón Alcarria, Diego Sánchez de Rivera, and Alvaro Sánchez-Picot. Enhancing process control in industry 4.0 scenarios using cyber-physical systems. *JoWUA*, 7(4):41–64, 2016.

[72] Teodora Sanislav, George Mois, Silviu Folea, Liviu Miclea, Giulio Gambardella, and Paolo Prinetto. A cloud-based cyber-physical system for environmental monitoring. In *2014 3rd Mediterranean Conference on Embedded Computing (MECO)*, pages 6–9. IEEE, 2014.

[73] 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.

[74] Sandeep Nair Mittal Sudip Joshi Anupam Sowmya Ramapatruni, Narayanan and Joshi Karuna. Learning behaviors in a smart home using hmm. In *2019 IEEE 5th IEEE International Conference on Big Data Security on Cloud (BigDataSecurity 2019)*. IEEE, 2019.

[75] Darlene Storm. Hack to steal cars with keyless ignition: Volkswagen spent 2 years hiding flaw.

[76] Adrian Taylor, Sylvain Leblanc, and Nathalie Japkowicz. Anomaly detection in automobile control network data with long short-term memory networks. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 130–139. IEEE, 2016.

[77] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. Canauth-a simple, backward compatible broadcast authentication protocol for can bus. In *ECRYPT Workshop on Lightweight Cryptography 2011*, 2011.

[78] 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.

[79] Stefan Wiesner, Eugenia Marilungo, and Klaus-Dieter Thoben. Cyber-physical product-service systems–challenges for requirements engineering. *International journal of automation technology*, 11(1):17–28, 2017.

[80] Marko Wolf, André Weimerskirch, and Thomas Wollinger. State of the art: Embedding security in vehicles. *EURASIP Journal on Embedded Systems*, 2007(1):074706, 2007.

[81] Kim Zetter. *Countdown to Zero Day: Stuxnet and the launch of the world's first digital weapon*. Broadway books, 2014.

[82] Xiaoqiang Zhang, Pingzhi Fan, and Zhongliang Zhu. A new anomaly detection method based on hierarchical hmm. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 249–252. IEEE, 2003.

[83] Yian Zhou, Zhen Mo, Qingjun Xiao, Shigang Chen, and Yafeng Yin. Privacy-preserving transportation traffic measurement in intelligent cyber-physical road systems. *IEEE Transactions on Vehicular Technology*, 65(5):3749–3759, 2016.

[84] Tobias Ziermann, Stefan Wildermann, and Jürgen Teich. Can+: A new backward-compatible controller area network (can) protocol with up to 16x higher data rates. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1088–1093. European Design and Automation Association, 2009.