IoT4CPS – Trustworthy IoT for CPS

FFG - ICT of the Future

Project No. 863129

Deliverable D4.6

Laboratory demonstrator of IoT anomaly detection and Threat Intelligence

The IoT4CPS Consortium:

AIT – Austrian Institute of Technology GmbH
AVL – AVL List GmbH
DUK – Donau-Universität Krems
IFAT – Infineon Technologies Austria AG
JKU – JK Universität Linz / Institute for Pervasive Computing
JR – Joanneum Research Forschungsgesellschaft mbH
NOKIA – Nokia Solutions and Networks Österreich GmbH
NXP – NXP Semiconductors Austria GmbH
SBA – SBA Research GmbH
SRFG – Salzburg Research Forschungsgesellschaft
SCCH – Software Competence Center Hagenberg GmbH
SAGÖ – Siemens AG Österreich
TTTech – TTTech Computertechnik AG
IAIK – TU Graz / Institute for Applied Information Processing and Communications
ITI – TU Graz / Institute for Technical Informatics
TUW – TU Wien / Institute of Computer Engineering
XNET – X-Net Services GmbH

For more information on this document or the IoT4CPS project, please contact:

Mario Drobics, AIT Austrian Institute of Technology, mario.drobics@ait.ac.at

The IoT4CPS project is partially funded by the "ICT of the Future" Program of the FFG and the BMVIT.

Document Control

Title: Laboratory demonstrator of IoT anomaly detection and Threat Intelligence

Type: Public

Editor(s): Faiq Khalid (TUW)

E-mail: faiq.khalid@tuwien.ac.at

Author(s): Faiq Khalid (TUW), Muhammad Shafique (TUW), Arndt Bonitz (AIT)

Doc ID: IoT4CPS-D4.6

Amendment History

Version	Date	Author	Description/Comments
V0.1	22.09.2020	Arndt Bonitz	Initial document version prepared
V0.2	23.09.2020	Arndt Bonitz	Added additional information about demonstrator
V0.3	03.11.2020	Wolfgang Herzner	Internal review
V0.4	04.11.2020	Arndt Bonitz	Changes and additions after the first review
V0.5	30.11.2020	Faiq Khalid	Simulation setup for demonstrating defense the secure VANET
V0.6	03.12.2020	Heinz Weiskirchner	2nd internal review
V1.0	09.12.2020	Faiq Khalid	Final Version

Legal Notices

Content

Abbreviations

API	Application Programming Interface
AES	Advanced Encryption Standard
ARM	Advanced RISC Machine
CoAP	Constrained Application Protocol
COTS	commercial off-the-shelf
CPSs	Cyber Physical Systems
DoS	Denial of Service
DPR	Dynamic Partial Reconfiguration
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
HT	Hardware Trojan
HK	Human Knowledge
IIoT	Industrial Internet of Things
ILANG	Intermediate Language
ISTQB	International Software Testing Qualifications Board
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
PSL	Property Specification Language
RNG	Random Number Generator
R/S	Rescaled Range
RTL	Register-Transfer Level
SDL	Security Development Lifecycle
SCA	Side-Channel Analysis
SMT	Satisfiability Modulo Theory
SoC	System on Chip
SSA	Static Single Assignment
SUT	System Under Test
TLS	Transport Layer Security
TRNG	True Random Number Generators
V2V	Vehicle-to-Vehicle
IoT	Internet of Things
CPS	Cyber-Physical Systems
VANET	Vehicular Ad hoc Network
ITS	Intelligent Transportation System
RSUs	Road-Side Units
OBUs	On-Board Units
SeVANET	Secure Vehicular Ad hoc Networks

BSM	Basic Safety Message
APP Layer	Application Layer
PHY Layer	Physical Layer
CFM	Car-Following Model

Executive Summary

In safety critical IoT CPS environments, like an intelligent transport system, defence in depth against cyber-threats is crucial. Towards this goal, in this deliverable, we present the two laboratory demonstrators to illustrate the novel anomaly detection approaches and communication-based threat detection approaches. The first demonstrator, based on COTS technology, introduced in this deliverable aims to show how critical devices can be secured via novel approaches in anomaly detection, even when state-of-the-art security measures fail. The second demonstrator shows the proof-of-concept of our proposed technique, SeVANET, for successfully identifying several security threats, i.e., leakage, denial-of-service. This demonstrator shows the effectiveness of SeVANET on Vehicular Ad hoc Network (VANET) running on the state-of-the-art simulator (i.e., VANET Simulink Toolbox) against different communication attacks on VANET, e.g., Denial-of-Service.

1 Introduction

In today's ICT landscape, systems become increasingly interconnected and thus more and more vulnerable to cybersecurity threats. Especially with the emergence of IoT, the attack surface towards malicious attacks is tremendously increased. However, traditional software or hardware security measures cannot adequately address these security vulnerabilities due to the enormous amount of data, interdependencies between the physical and cyber worlds, and energy/resource constraints in the CPS. In order to tackle this issue, it is equally important to deploy defence in depth security measures. This can help to maintain a high level of security even if one individual measure is compromised or fails.

In this deliverable, we describe two laboratory demonstrators to cover the wider range of attack surfaces.

1. The first laboratory demonstrator combining state-of-the-art security measures, such as authentication and encryption, with a novel approach for anomaly detection.
2. The second demonstrator is a simulator based on the state-of-the-art VANET Simulink Toolbox[1], that combines SeVANET[2] with the state-of-the-art component in VANET.

In chapter 1, this deliverable summarizes one approach for securing IoT devices, that was undertaken within the IoT4CPS project. Chapter 2 describes the basic concept of the demonstrator. In chapter 3, the set-up of the laboratory demonstrator is outlined. Chapter 4 shows the results and chapter 5 summarizes the deliverable.

[1] Le Wang (2020). VANET Toolbox: A Vehicular Network Simulator based on DES (https://www.mathworks.com/matlabcentral/fileexchange/68437-vanet-toolbox-a-vehicular-network-simulator-based-on-des), MATLAB Central File Exchange. Retrieved December 1, 2020.

[2] SeVANET is our novel technique that leverages the statistical modelling of a communication pattern (i.e., Hurst exponent) to generate a single parameter-based unique signature for a particular type of communicating command. Besides, we design low-overhead hardware modules to extract the statistical parameter of communication patterns during run-time. These signatures extracted during design-time are compared with extracted signatures during run-time to identify anomalous communication behaviour.

2 Demonstrator Concept

IoT, CPS and Industry 4.0 lead to an increasing interconnection between the physical and digital world. In many scenarios, there is no central understanding of complete systems: experts only have knowledge for layers, components, or specific technologies. For example, an expert might know how to configure and program a manufacturing robot but has no insight into the company network structure to which the robots are connected, nor into the setup of the office environment in which the robots are programmed. In many cases, this mix of technologies can be monitored and controlled only with a mix of security solutions. Each of these security components offers numerous configurations or operation modes, much more than humans can understand or comprehend (but only a few might be relevant). Therefore, numerous unknown attack modes (vectors) exist, making efforts in defense in depth crucial. If one security measure fails, there must be others to mitigate the risks of an adversary successfully attacking the whole system. This is especially true for safety-critical CPS systems, since here a malfunction can lead to great impacts, including loss of life.

2.1 Demonstrator 1

This laboratory demonstrator aims to show how a failure in one security measure can potentially impact the safety and security of a CPS system, but also shows how other, novel approaches can detect and potentially mitigate the effects of a successful attack. Figure 1 shows the basic concept of this demonstrator. The demonstrator consists of two RaspberryPi SOCs, a client that can run on any Microsoft Windows machine, a robot arm and second generic MQTT client that can run on any device (in this case an Android smartphone). All devices are connected via a WiFi network, in this case hosted by the SOC #1 RaspberryPi. Here, also a MQTT (MQ Telemetry Transport) broker is running. MQTT is a lightweight publish/subscribe messaging protocol and is widely used for machine-to-machine communication in the world of IoT. With MQTT, clients do not communicate directly, but “publish” their messages to a “topic” on a broker. Other clients can subscribe to this topic an consume the messages. While this is ideal for establishing a robust and simple communication scheme, the subscribed client has no way of knowing who sent out the messages in the first place. This means, a CPS device might also act upon erroneous or malicious commands received via MQTT. In order to mitigate this issue, there are MQTT implementations allowing for client authentication and encrypted communication. However, many MQTT brokers in the wild are not making use of this important security measure[3,4]. This can either be the result of a lack of knowledge and awareness, or configuration mistakes. To do justice to reality, MQTT broker allows to be run in a secure and unsecured mode:

- Unsecured: The broker accepts any client with no authentication needed, the communication itself is not encrypted.
- Secured: The broker only accepts clients authenticated via a client certificate; the communication is encrypted via TLS/SSL.

[3] https://blog.avast.com/mqtt-vulnerabilities-hacking-smart-homes

[4] Andy, Syaiful, Budi Rahardjo, and Bagus Hanindhito. "Attack scenarios and security analysis of MQTT communication protocol in IoT system." 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI). IEEE, 2017.

SOC #1 also features a server management client, which allows the user to easily enable or disable the MQTT security features. SOC #2 controls the robotic arm. Here, a MQTT client directly translates MQTT messages in commands for the robotic arm. Furthermore, as a second line of defence, it features the AIT AMiner[5], which provides automatic event correlation for incident detection. AMiner parses and interprets all incoming commands and evaluates them against a previously learned "normal system behavior". This enables the detection of an atypical use of the system, and consequently the detection of attacks from a malicious actor.

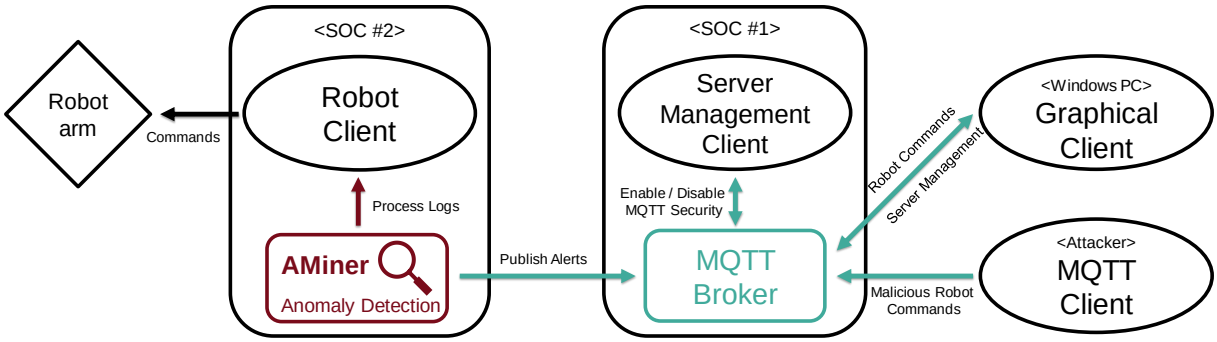

Figure 1 High Level Concept

2.2 Demonstrator 2

In the emerging ITS, VANET is one of the prominent systems to manage V2V communication and V2I communication. Typically, it includes several entities, like RSUs, OBUs, vehicles, communicating with each other using wireless communication protocols or physical communication within a vehicle or RSUs. Therefore, to emulate these real-world scenarios, we used the state-of-the-art VANET Simulink toolbox to generate different attack and defence scenarios. This laboratory simulator aims to demonstrate the effectiveness of the SeVANET under the real-world scenario of V2V communication. Figure 2 shows the basic concept of the simulation setup for designing secure VANTE. This setup consists of the following components:

1. **Communication Analyzer:** It first simulates the given scenario, then it analyses and records each communication channel's communication pattern.
2. **Hurst Exponent Estimation:** After recording the communication for a limited time, it computes the Hurst exponent of each communication trace to model the respective statistical behaviour.
3. **Modification of VANET library:** For the run-time Hurst Exponent estimation, we designed the hardware component. Then, we embedded the hardware components in the respective communication modules of the VANET library.

[5] https://aecid.ait.ac.at/

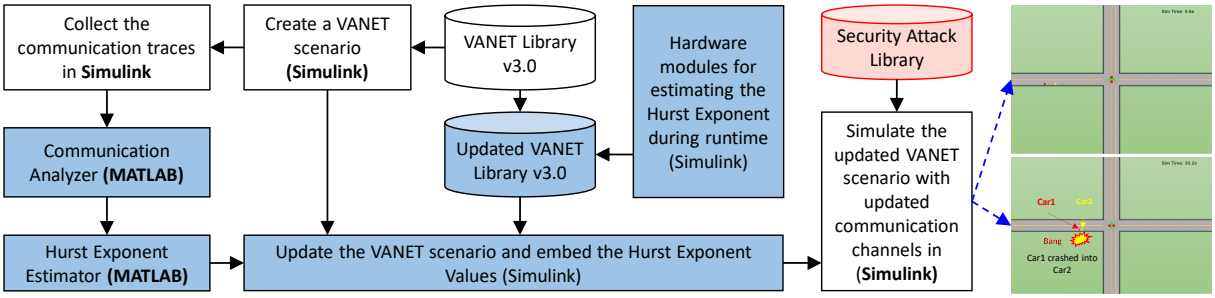

Figure 2: Overview of the simulation setup based on the state-of-the-art VANET Simulink Toolbox.

2.2.1 Hardware Modules for estimating the Hurst Exponent

To extract the statistical parameters during the run-time, we design the hardware modules for computing the Hurst exponent. Towards this, we used one of the most commonly used methods, i.e., the R/S method. The reason behind choosing this method is that it requires a smaller number of observations to estimate the Hurst exponent.

Algorithm 1: Estimation of Hurst Exponent

Input:
n: Number of the observations = 512
$X[0:n-1]$: number of communication packets per observation
Generate three computation blocks using $X[0:n-1]$
Computation Block 1:
Compute the mean and the standard deviation of $X[0:n-1]$
Compute the mean centered series: $h[0:n-1] = X[0:n-1] - M$
Compute cumulative deviation by summing up the mean centered values:
$Y[0:n-1] = \sum_{i=0}^{0} h[i], \sum_{i=0}^{1} h[i], ..., \sum_{i=0}^{n-1} h[i]$
Compute the Range (R) of $Y[0:n-1]$
Compute R/S
Compute $E(R/S)_0$
Computation Block 2:
Compute the mean (M) and standard deviation (S) of $X[0:(n/2)-1]$ and $X[n/2:n-1]$
Compute the mean centered series:
$h_1[0:(n/2)-1] = X[0:(n/2)-1] - M$ and $h_2[n/2:n-1] = X[n/2:n-1] - M$
Compute cumulative deviation by summing up the mean centered values:
$Y[0:(n/2)-1] = \sum_{i=0}^{0} h_1[i], \sum_{i=0}^{1} h_1[i], ..., \sum_{i=0}^{(n/2)-1} h_1[i]$
$Y[n/2:n-1] = \sum_{i=n/2}^{(n/2)} h_2[i], \sum_{i=n/2}^{(n/2)+1} h_2[i], ..., \sum_{i=n/2}^{n-1} h_2[i]$
Compute the Range (R) of $Y[0:(n/2)-1]$ and $Y[n/2:n-1]$
Compute $(R/S)_1$ and $(R/S)_2$
Compute $E(R/S)_1 = \frac{(R/S)_1+(R/S)_2}{2}$
Computation Block 3:
Compute the mean (M) and standard deviation (S) of $X[0:(n/4)-1]$, $X[n/4:(n/2)-1]$, $X[n/2:(3n/4)-1]$ and $X[3n/4:n-1]$
Compute the mean centered series:
$h_1[0:(n/4)-1] = X[0:(n/4)-1] - M$, $h_2[n/4:(n/2)-1] = X[n/4:(n/2)-1] - M$,
$h_3[n/2:(3n/4)-1] = X[n/2:(3n/4)-1] - M$ and $h_4[3n/4:n-1] = X[3n/4:n-1] - M$
Compute cumulative deviation by summing up the mean centered values:
$Y[0:(n/4)-1] = \sum_{i=0}^{0} h_1[i], \sum_{i=0}^{1} h_1[i], ..., \sum_{i=0}^{(n/4)-1} h_1[i]$
$Y[n/4:(n/2)-1] = \sum_{i=n/4}^{n/4} h_2[i], \sum_{i=n/4}^{(n/4)+1} h_2[i], ..., \sum_{i=n/4}^{(n/2)-1} h_2[i]$
$Y[n/2:(3n/4)-1] = \sum_{i=n/2}^{n/2} h_3[i], \sum_{i=n/2}^{(n/2)+1} h_3[i], ..., \sum_{i=n/2}^{(3n/4)-1} h_3[i]$
$Y[3n/4:n-1] = \sum_{i=3n/4}^{3n/4} h_4[i], \sum_{i=3n/4}^{(3n/4)+1} h_4[i], ..., \sum_{i=3n/4}^{n-1} h_4[i]$
Compute the Range (R) of $Y[0:(n/4)-1]$, $Y[n/4:(n/2)-1]$, $Y[n/2:(3n/4)-1]$ and $Y[3n/4:n-1]$
Compute $(R/S)_1$, $(R/S)_2$, $(R/S)_3$ and $(R/S)_4$
Compute $E(R/S)_2 = \frac{(R/S)_1+(R/S)_2+(R/S)_3+(R/S)_4}{4}$
Hurst Exponent:
Compute $E(R/S) = \frac{E(R/S)_0+E(R/S)_1+E(R/S)_2}{3}$
Compute Hurst exponent $H = 0.37 \times logE(R/S)$

1. After establishing a communication channel, it observes and stores the number of packets per clock cycle, as denoted by X in Algorithm 1. In SeVANET, we set the number of observations to 512. The reason behind choosing this value is that for the fast convergence of Hurst exponent, the number of observations should be higher than 300, with minimum memory overhead. Note, to estimate the Hurst

exponent. We used one of the most commonly used methods, i.e., the R/S method. The reason behind choosing this method is that it requires a fewer number of observations to estimate the Hurst exponent. We implemented a state-of-the-art modified non-restoring algorithm for computing the square root function in standard deviation.

2. To estimate the Hurst exponent, it is important to take the average value of R/S values using the different data distribution obtained from the same data. Therefore, we propose to use three computational blocks, as shown in algorithm 1 and Figure 3. The data flow and hardware modules in each computational block are the same, but the sizes of hardware modules are different. Computational block 1 uses the complete data for estimating the R/S value. Computational block 2 divides the data into two equal parts. Then this block estimates the respective R/S values using each half data and takes the average to estimate the R/S value. Similarly, computational block 3 repeats the same procedure, but it divides the data into four equal parts.
3. In each computational block, the first step is to compute the mean value (M) and standard deviation (S) of the input data series X. The hardware modules to compute the mean value consists of "n" 64-bit adders, one 9-bit shifter, and one output register, as shown in Figure 3. The hardware module for computing the standard deviation consists of "n" 64-bit subtractors, "n" 64-bit multipliers, one 9-bit shifter, one module to compute the square root, and one output register.
4. After computing the mean and standard deviation, each computational block computes the mean-centered data series (H) by subtracting the mean value from each input data series (X), as shown in algorithm 1. Then, it computes the cumulative deviation (Y) by summing up the mean-cantered data series (H) and computes the magnitude range (R) of the cumulative deviation (Y). Finally, it computes the R/S using a 64-bit divider. Note, there is no extra hardware for the mean-centered series, and each computational block uses the values from the standard deviation module. The hardware module of computing Y consists of one 64-bit accumulator, as shown in Figure 3, and the hardware module computing R consists of two comparators, two multiplexers, and one 64-bit subtractor.
5. Finally, it computes the R/S by taking the average of R/S values computed from each computational block. Finally, the average R/S value is used to calculate the Hurst exponent using $H = 0.37 \times \log_{10}(\text{the average value of R/S})$, as shown in algorithm 1 and Figure 3.

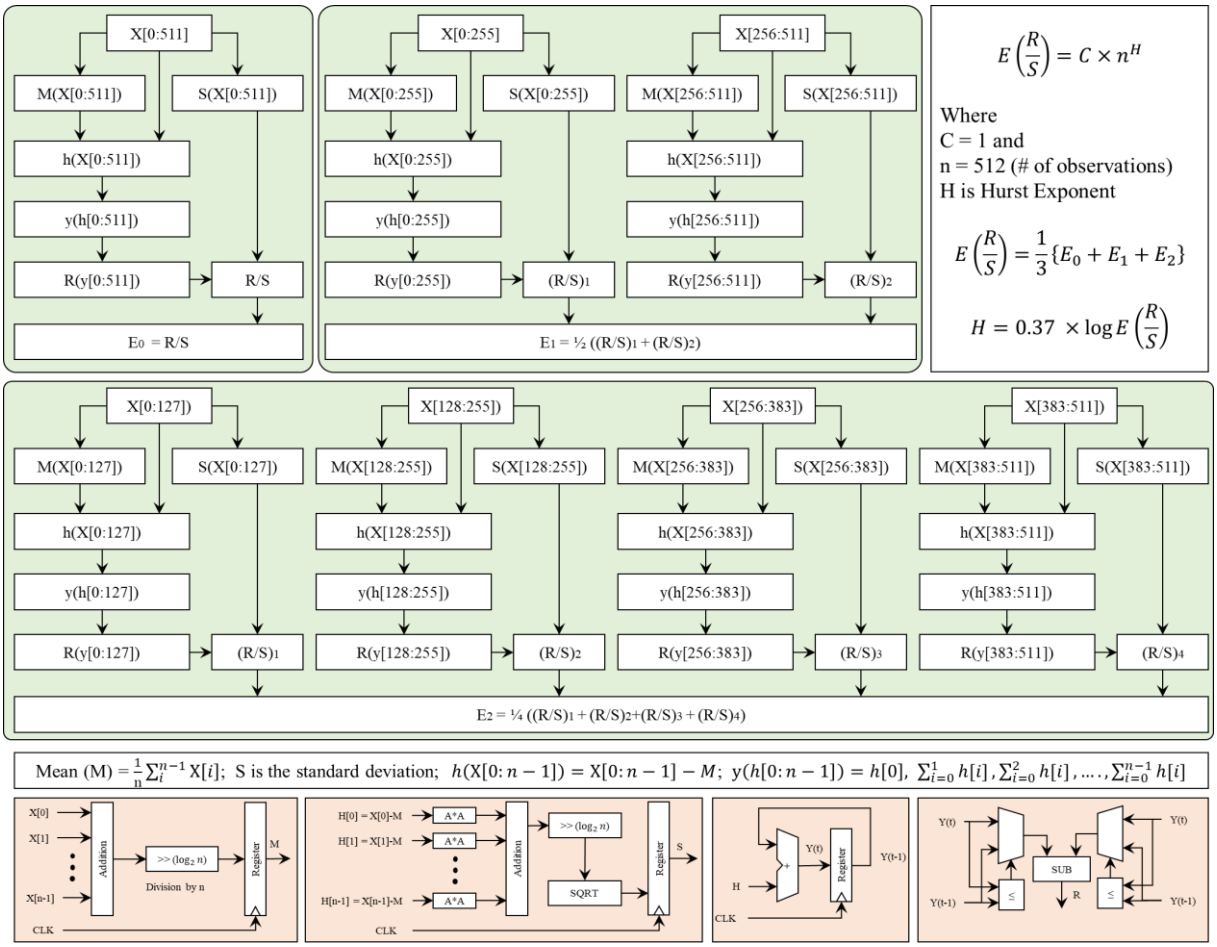

Figure 3: Data flow of the hardware module to compute the Hurst exponent.

2.2.2 Modification of VANET Library

In order to simulate the effect of statistical modelling of the V2V and V2I communication, we have modified the communication blocks of the state-of-the-art MATLAB Simulink library, i.e., VANET toolbox library v3.0, as shown in Figure 4. In each communication module, we inserted a hardware module for estimating the Hurst exponent (see Figure 3) for each communication channel. VANET Simulink toolbox contains the following major layers:

1. **Application Layer (APP layer):** This layer generates messages including Basic Safety Message (BSM) and Lane Changing Message. Moreover, this layer's mobility models include the car-following model (CFM) and lane-changing model (LCM), and it helps stimulate the braking feature.
2. **MAC layer:** It includes the implementation of Enhanced Distributed Channel Access (EDCA) according to IEEE 802.11p, which simulates the communication between the application layer to the physical layer.
3. **Physical Layer (PHY layer):** It includes a two-ray ground reflection model and an AWGN channel based on WAVE/DSRC standards (i.e., IEEE 802.11a) to simulate the wireless communication.

Although estimating the Hurst exponent of each channel increases the threat detection accuracy, it increases the computational and power overhead. Therefore, we propose to use state-of-the-art topologies[6] for deploying the

[6] F. Khalid, S. R. Hasan, O. Hasan and F. Awwad, „Runtime hardware Trojan monitors through modeling burst mode communication using formal verification," In Integration, vol. 61, 2018, pp. 62-76.

hardware modules for estimating the Hurst exponent. Based on the design constraints, the designer can choose an appropriate setup of the Hurst exponent modules by using the proposed methodologies.

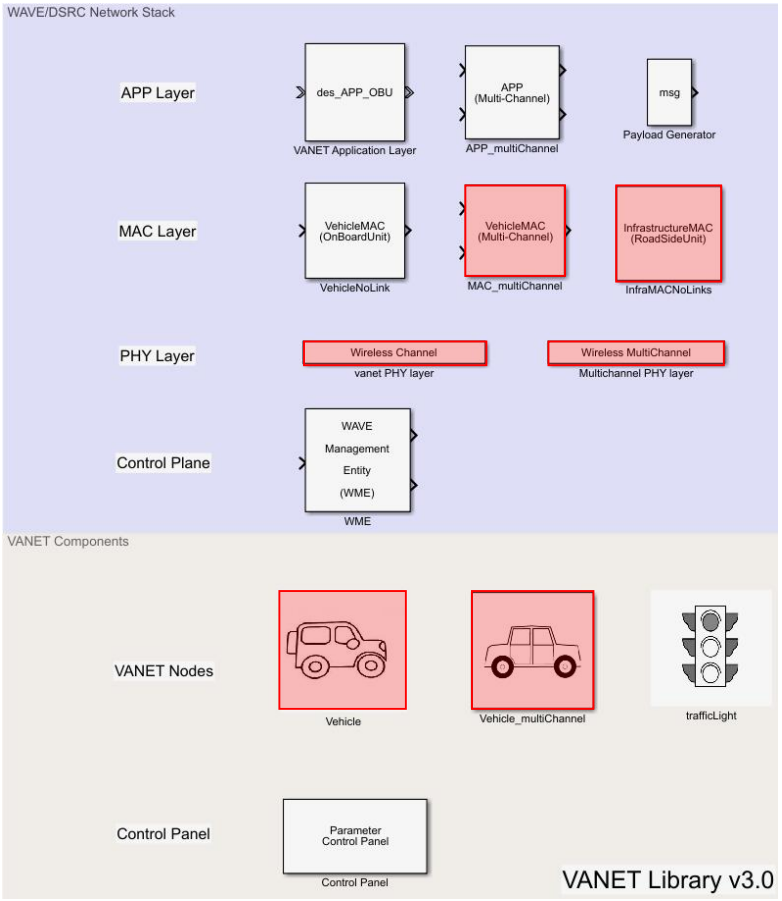

Figure 4: The screenshot of the VANET library from the state-of-the-art VANET toolbox from MATLAB Simulink. The red-highlighted boxes represent those modified by SeVANET (embedded the hardware for Hurst exponent (see Figure 3) into the respect communication modules).

3 Laboratory Demonstrator

The laboratory demonstrator reflects a scenario of a CPS manufacturing system, controlled via MQTT from a standard Windows workstation. Figure 5 shows the main components of the demonstrator in action: both RaspberryPI SOCs and the robot arm lifting a block with the pneumatic pump.

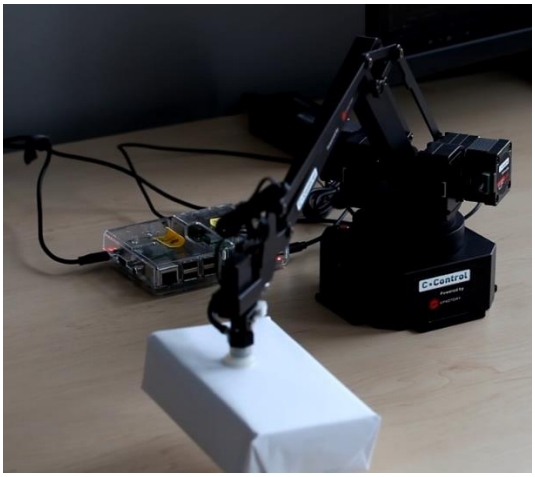

Figure 5 The laboratory demonstrator

3.1 Set-up

This section outlines the setup of the laboratory demonstrator and the components used to imitate a basic industrial environment setup.

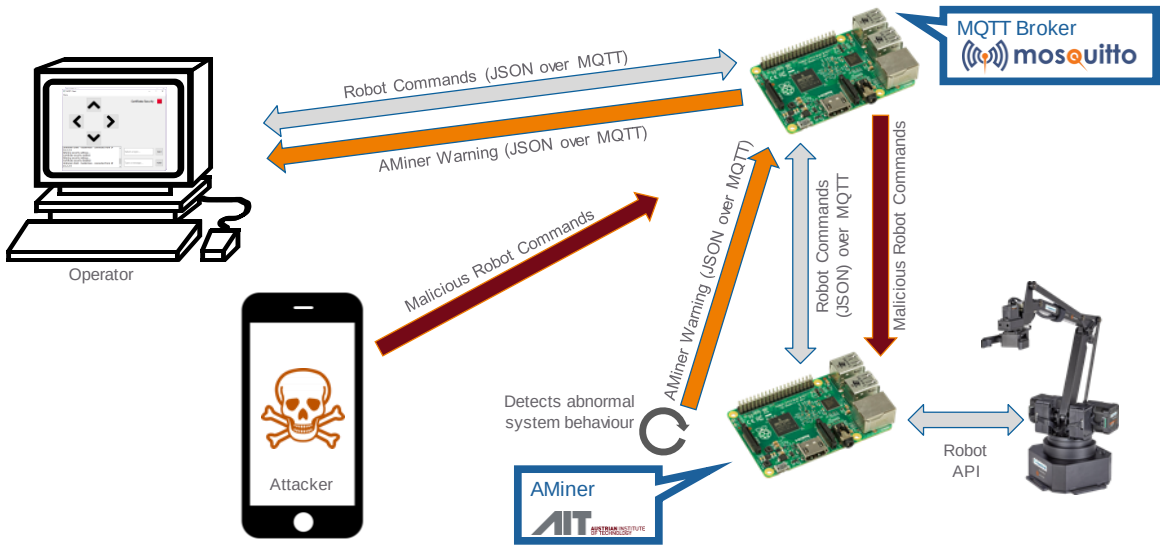

Figure 6 Setup laboratory demonstrator for securing IoT CPS devices

Figure 6 shows that setup with following components in place:

1. Robotic Arm: C-Control Robotic arm gripper set CCR-45.
   Connected via USB 2.0 to SOC #2. Controlled via Python 3 API.
2. SOC #1 & #2: Raspberry Pi 3 Model B, running Raspberry Pi OS (32-bit)
3. Operator PC: Standard desktop PC hardware, running Windows 10 Professional.
   Graphical Client implemented in Python 3 / Qt.

4. Smartphone

   Regular Android 9.0 smartphone, with MQTT.fx[7] client installed.

3.2 User interface

Figure 7 depicts the Windows User Interface for interacting with the laboratory demonstrator.

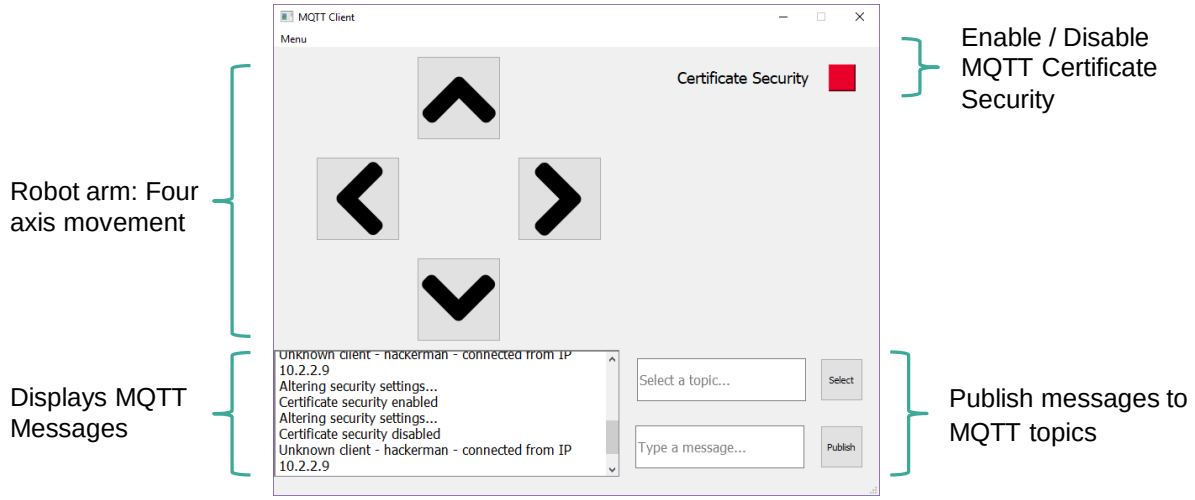

Figure 7 Windows User Interface

Figure 8 shows an Android smartphone with the MQTT.fx client. The robot MQTT commands for left motion, drop a payload and pump interaction already configured here.

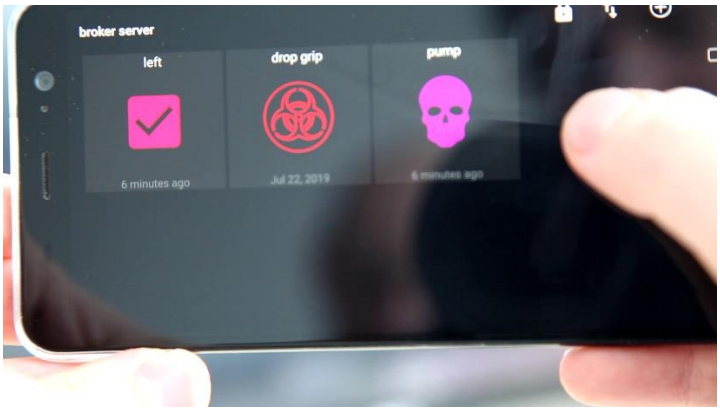

Figure 8 MQTT.fx Android client

[7] https://mqttfx.jensd.de/

4 Experimental Results

4.1 Demonstrator 1

As a simple example of an industrial process, a sequence of commands was developed that instructed the robotic arm to move one block with the suction pump from one side to the other, and vise versa. This sequence was then "learned" by AMiner in order to establish a normal system behavior. After learning the normal system behavior, i.e. the usage pattern of the robot arm, the MQTT security was deactivated. As a consequence, the "attacker" could connect to the now unsecured MQTT broker via the MQTT.fx Android client (Figure 8). Subsequently, the attacker could disrupt the normal operation – for example by issuing the "pump" command, which turns on and off the suction pump.

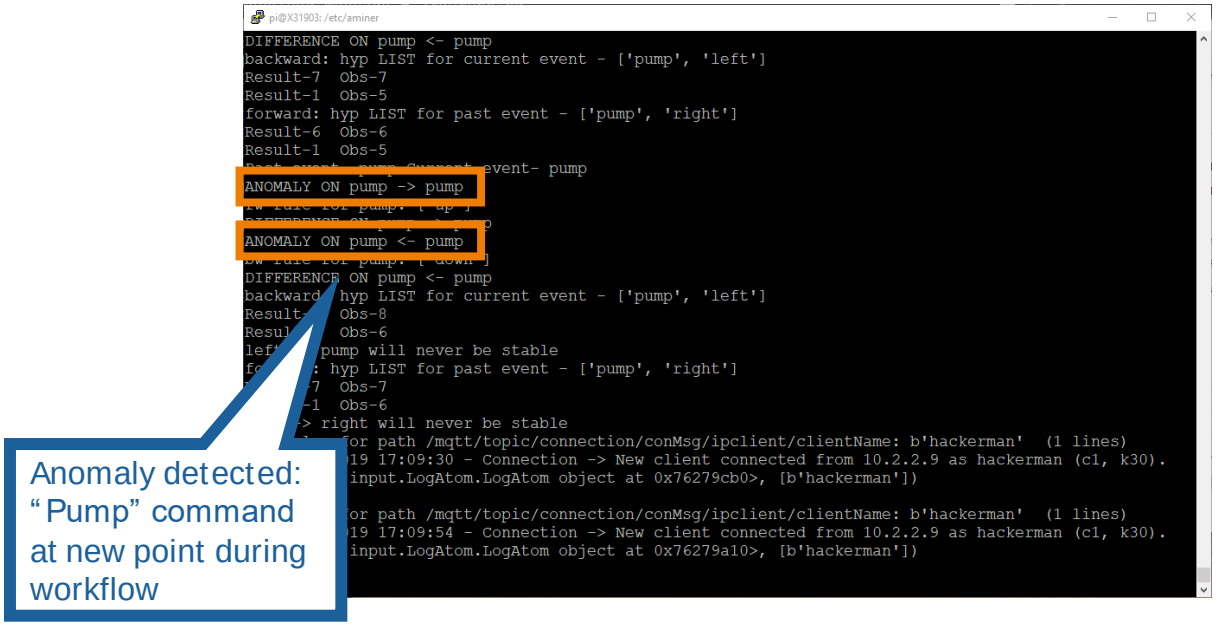

Figure 9 Console output of AMiner

As shown in Figure 9, AMiner managed to detect the manipulation attempts by the attacker. For example, a new anomaly was detected, since the "pump" command occurred at a new point during the workflow. Other commands that were in contradiction to the learned system behavior were also detected.

4.2 Demonstrator 2

To illustrate the effectiveness of the SeVANET, we implemented a proof of concept scenario, i.e., communication between two cars approaching an intersection while maintaining a safe distance.

4.2.1 Experimental Setup

There are two autonomous vehicles in this scenario, i.e., Car1 (red dot) and Car2 (Yellow dot), with different features like initial speed, acceleration, initial position, as shown in Figure 10. These cars are also communicating with each other and with a traffic light controller using a wireless communication channel. Moreover, in this experiment, the attacker interrupts the communication between the road-side unit (i.e., traffic light controller) and Car1 and injects false information about the traffic signal status. As a result of the attack, Car1 fails to

decelerate and eventually crashes into Car2. Note, SeVANET is only an attack detection technique, but we used the most common mitigation technique for the realistic scenario, i.e., restart the potentially compromised V2V or V2I communication.

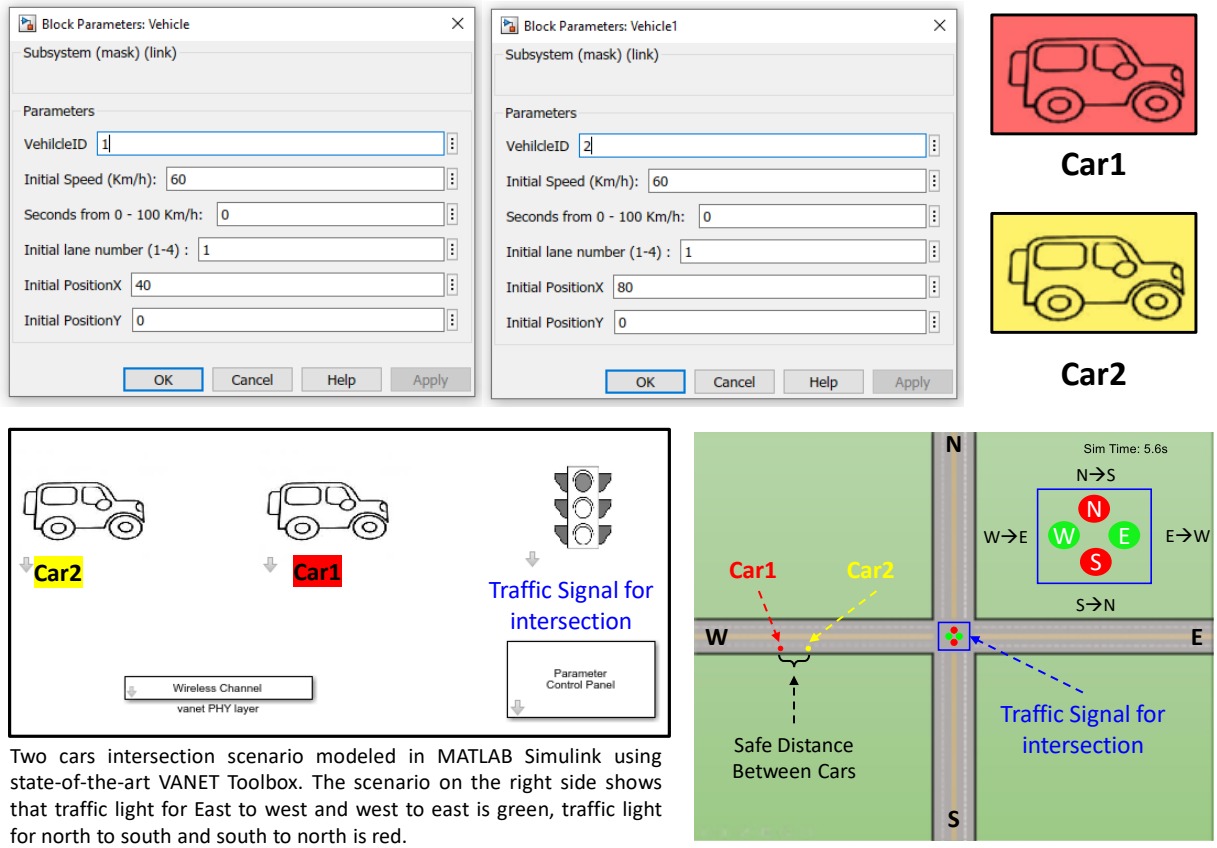

Figure 10: An example scenario, i.e., communication between two cars approaching an intersection while maintaining a safe distance, to demonstrate the effectiveness of SeVANET.

4.2.2 Key Observations and Discussion

To show the effectiveness of the proposed solution (SeVANET), we simulated the above-mentioned scenario without any defense and with the SeVANET. The experimental results for these two cases are shown in Figure 11 (without any defense) and Figure 12 (with SeVANET), respectively. The results in Figure 11 show that when there is no defense mechanism in VANET, a state-of-the-art DoS attack is launched on Car1. This attack forces the Car1 to fail the deceleration that leads to the devastated crash with Car2. However, the deployment of SeVANET in the wireless communication channel of both Cars nullifies the DoS attack. Therefore, both cars have crossed the intersection without any devastated crash, as shown in Figure 12. This proof-of-concept shows that monitoring the statistical modeling of the inter/intra communication between different vehicles and RSUs can successfully identify the potential attack. Hence, our proposed solution (SeVANET) can be used as an attack detection schemes along with state-of-the-art prevention technique.

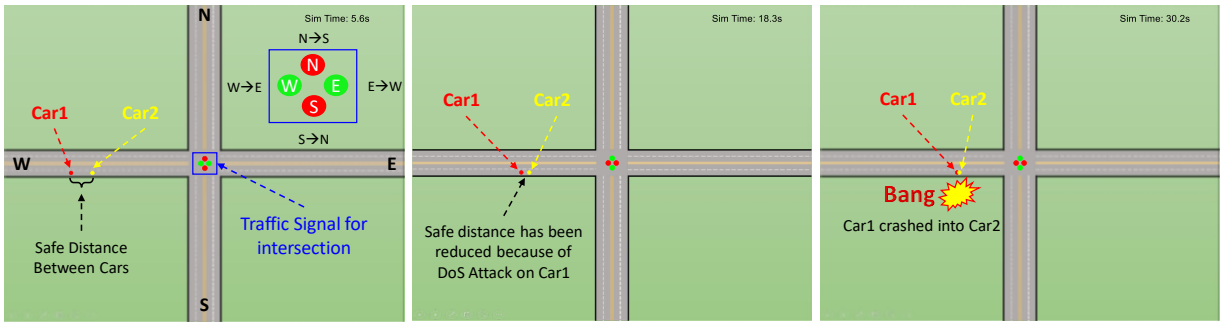

Figure 11 Demonstrator results show that the DoS attack to crash the car1 has been successfully executed in the absence of any defense mechanism. In this experiment, the attacker interrupts the communication between the road-side unit (i.e., traffic light controller) and Car1 and injects false information about the traffic signal status. As a result of the attack, Car1 fails to delectate and eventually crashes into Car2.

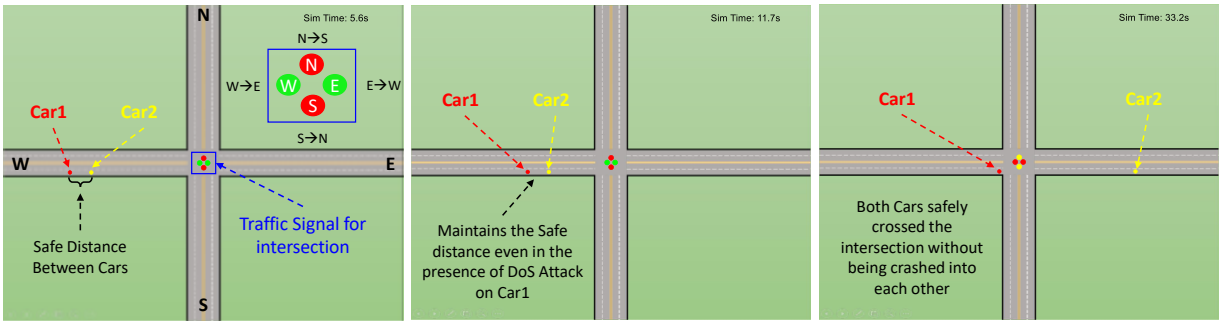

Figure 12 Demonstrator results show that the DoS attack to crash the car1 has been successfully nullified by the proposed SeVANET. As a result, both cars crossed the intersection with being crashed into each other.

5 Conclusion

The first lab demonstrator showed that it is feasible to establish defense in depth providing redundancy in the event a security control failing within an IoT CPS environment. It shows in an easily understandable manner, that adding additional, novel security controls can improve security substantially. The lab demonstrator was successfully introduced to a wider audience at the European Robotics Forum 2020 in Malaga, Spain and was well received. A demonstration video showing the set-up, attacks and anomaly detection is available on the AIT YouTube channel[8].

The simulation setup in the second demonstrator used the state-of-the-art VANET Simulink toolbox to show the feasibility of monitoring the run-time statistical model of the inter/intra communication in VANET for detecting the potential attack. Based on its experimental results, we can conclude that the proposed solution can be used to complement the state-of-the-art mitigation/defense techniques for securing VANET.

In conclusion, both demonstrators show novel approaches on how to tackle important security issues that arise, when combining CPS and IoT.

[8] https://youtu.be/TeW6YHvpqtc