Exploring the consequences of cyber attacks on Powertrain Cyber Physical Systems

# Exploring the consequences of cyber attacks on Powertrain Cyber Physical Systems

Dario Stabili*, Raffaele Romagnoli†, Mirco Marchetti*, Bruno Sinopoli‡ and Michele Colajanni§

* University of Modena and Reggio Emilia
† Carnegie Mellon University
‡ Washington University in St. Louis
§ University of Bologna

Emails: *{dario.stabili, mirco.marchetti}@unimore.it, †rromagno@andrew.cmu.edu
‡bsinopoli@wustl.edu, §michele.colajanni@unibo.it

*Abstract*—This paper proposes a novel approach for the study of cyber-attacks against the powertrain of a generic vehicle. The proposed model is composed of a generic Internal Combustion engine and a speed controller, that communicate through a Controller Area Network (CAN) bus. We consider a threat model composed of three representative attack scenarios designed to modify the output of the model, thus affecting the rotational speed of the engine. Two attack scenarios target both vehicle sensor systems and CAN communication, while one attack scenario only requires injection of CAN messages. To the best of our knowledge, this is the first attempt of modeling the consequences of realistic cyber attacks against a modern vehicle.

*Index Terms*—cyber attacks, cyber-physical systems, powertrain, controller area network, control system.

## I. INTRODUCTION

Since the introduction of microcontrollers in modern vehicles, car manufacturers began to push more and more features to improve both safety and driving comfort. These features are deployed on microcontrollers that are part of the vehicle network and called Electronic Control Units (ECUs), which are connected to the mechanical parts of the vehicle and communicate between each others through means of different communication protocols. The most deployed communication protocol is the Controller Area Network (CAN), developed by Bosch GmbH in the early '90 [1]. Despite the CAN bus is one of the most deployed networking protocols in modern vehicles it does not provide any security guarantees. Security researchers demonstrated that it is possible to hijack a vehicle by injecting maliciously forged messages on the CAN bus, and published these results on technical reports and white papers [2], [3]. These attacks are achieved by exploiting the *drive-by-wire* capabilities of modern vehicles, a feature that enables the control of the driving system through messages sent over the CAN bus. An example of the *drive-by-wire* capabilities of modern vehicle is represented by the cruise control, which is activated to preserve the speed of the vehicle to reduce fuel consumption and therefore emissions.

Despite these systems are developed for comfort and safety purposes, the fundamentals required for their deployment pave the way to targeted attacks, jeopardizing the safety of people inside and outside the vehicle. Since the first public demonstration of a remote attack to a modern connected vehicle [4], many cyber security researchers proposed different detection algorithms tailored for the in-vehicle communication networks to detect an ongoing cyber-attack [5], [6], [7]. However, current literature explored attacks and their relative countermeasures only focusing on a particular vehicle domain. As an example, the detection algorithms designed to detect cyber-attacks targeting the communication network of modern vehicles are tested only against attacks targeting these networks, while detection methods based on the analysis of the system state are only tested against attacks designed to modify the system state.

In this paper, we present a complete powertrain system of a generic vehicle composed of an internal combustion engine and speed controller, connected using a Controller Area Network (CAN) communication protocol. This system is used to analyze the consequences of cyber-attacks to the powertrain system targeting both the sensors and the communication network.

This paper has two main contributions. The first one is that the speed controller is designed and implemented by using the model of the engine instead of relying on a generic representation of the whole vehicle [8]. To the best of our knowledge, this is the first time that the speed control problem is addressed by proposing a solution based on the engine model. The second contribution of this work is the demonstration of the consequences of cyber-attacks to the engine model, considering different attack scenarios that can be deployed at both system and communication level. To the best of our knowledge, this is the first paper that addresses the cyber security of the engine model using both model and communication methodologies.

The rest of the paper is organized as follows. Section II presents an analysis of the current state-of-the-art with the methodology presented in this paper, while Section III shows the design of the powertrain system, composed of the engine and controller models and the CAN messages used for communication between the two models. Section IV introduces the

threat model considered in this paper, and its effects on the engine model are presented in Section V. Final remarks and future work are outlined in Section VI.

## II. RELATED WORK

Modern vehicle systems are built by considering many different complex domains, that can be grouped as either mechanical [9], [10], electronic [11], or intercommunication domains [1], [12]. With the advent of remote connectivity, these systems transitioned from being isolated from outside communication to being interconnected with different communication technologies. The increasing adoption of novel Cooperative Intelligent Transport Systems (C-ITS) paved the way to the development of novel communication standards designed to meet the strict requirements of modern vehicles, targeting both safety and security of the communication [13], [14], [15], [16].

However, these novel external communication methods exposed internal and isolated networks to remote attacks, which take advantage of insecure networks to hijack the vehicle system [2], [3].

The vulnerabilities of the in-vehicle networks have already been addressed by cyber-security researchers. One of the first works presenting the vulnerabilities of a licensed vehicle is presented in [17], in which the authors experimentally evaluated the security of a modern vehicle. The work presented in [17] demonstrated the vulnerabilities of poorly-designed systems (such as the Adaptive Cruise Control) and the implications of using legacy communication protocols with no support to modern security techniques. Another similar work is presented in [4], in which the authors focused on the identification of a series of vulnerabilities to remotely hijacking the vehicle. Despite the detailed description of the methodology presented in [4] is focused on the steps required to gain remote access to the connected vehicle, the final goal of their work is to inject messages at CAN level, hence demonstrating how it is possible to hijack a connected vehicle via its drive-by-wire capabilities. In this paper we present first complete powertrain system of a modern vehicle, including both the engine model and CAN communication, that allows to investigate the consequences of cyber-attacks targeting both the internal communication network and the physical representation of the vehicle. Compared to the previous literature, the work presented in this paper presents a generalized approach for the analysis of the consequences of cyber-attacks to a generic engine model, instead of focusing on a particular vehicle.

Modern vehicles are cyber-physical systems (CPSs) [18]. CPSs are systems which show a tight connection between sensing, processing, control and communication to achieve goals that otherwise it would be impossible (e.g. advanced driver assistance system). This scenario offers many attack surfaces where cyber-security tools may not cover [19].

One example is the replay attack, where the adversaries replace the actual sensor measurements with pre-recorded ones, while driving the actual physical system to possible dangerous situations. To detect this kind of stealthy attack, system theory tools are needed since the attack is changing the information of the data and not the structure.

Classical system theory detection techniques are based on fault detection schemes that generate a residual value by comparing the real sensor measurements and the ones predicted by an observer. This residual value is later used for the anomaly detection task. In case of a replay attack, fault detection schemes are not able to detect it since the base values used for detection task are being corrupted. To overcome this limitation, an active detection method for replay attack on CPS has been proposed in [20].

Other interesting works are presented in [8], [21], in which a Denial-of-Service attack against a vehicle system is presented. While in [8] the authors focus on the analysis of the consequences of the missing input of a generic engine drive, the authors of [21] designed a predictive controller to overcome missing input. Both [8], [21] are based on a logical interruption of the input to the systems, which are represented by an approximation of the whole vehicle system instead of considering the actual mechanical parts composing the engine model. However, compared to both [8], [21], we showcase the consequences of a wider range of cyber-attacks, carried out on a model representing the actual mechanical composition of a modern internal combustion engine, and used a simulated communication network to study the realistic consequences of cyber-attacks targeting our system. Moreover, we adapted the replay attack presented in [20] to the automotive scenario to present its consequences on a novel scenario.

## III. POWERTRAIN MODEL

In this section the basic knowledge required for the understanding of this paper is provided. In Section III-A we describe the dynamical model of the IC engine, while in Section III-B we present the design of the controller deployed on the IC engine model. Finally, in Section III-C the basics of the Controller Area Network and on its design in our model are provided.

### A. Engine and Controller Models

We consider the mean-value model (MVM) of the sparkle-ignited (SI) engine, which consists of five interconnected sub-modules: throttle body, intake manifold, gas exchange, combustion and torque generation, and engine inertia. The nonlinear model that describes the dynamics of the SI engine is:

$$\frac{dp_m(t)}{dt} = \frac{R\theta_m}{V_d}(\dot{m}_\alpha(t) - \dot{m}_\beta(t)) \tag{1}$$

$$\frac{d\omega_e(t)}{dt} = \frac{1}{\theta_e}[T_e(t) - T_l(t)] \tag{2}$$

where $p_m$ is the intake manifold pressure, $\dot{m}_\alpha$ is the air-flow mass entering from the throttle valve, $\dot{m}_\beta$ is the air-flow mass going out from the intake manifold to the cylinders, $w_e$ is the engine speed, $T_e$ the torque generated by the engine, and $T_l$ the load torque. To use the aforementioned model for the design

of a controller, we need to identify the relationship between the two dynamics described in (1) and (2). Starting from (1) we have to characterize $\dot{m}_\alpha$ and $\dot{m}_\beta$. Assuming that the air is a perfect gas and the throttle is isenthalpic[1], $\dot{m}_\alpha$ is given by:

$$\dot{m}_\alpha(t) = \begin{cases} A_\alpha(t)\frac{p_a}{\sqrt{R\theta_a}}\frac{1}{\sqrt{2}} & \frac{p_m(t)}{p_a} \leq 0.5 \\ A_\alpha(t)\frac{p_a}{\sqrt{R\theta_a}}\sqrt{\frac{p_m(t)}{p_a}[1 - \frac{p_m(t)}{p_a}]} & else \end{cases}$$

(3)

where $A_\alpha$ is the throttle valve open area that is computed as follows:

$$A_\alpha(\alpha_{th}) = \frac{\pi d_{th}^2}{4}\left(\frac{\cos(\alpha_{th})}{\cos(\alpha_{th,0})}\right) + A_{th,leak}$$

(4)

$$\alpha_{th} = \alpha_{th,0} + \left(\frac{\pi}{2} - \alpha_{th,0}\right)u_\alpha$$

(5)

where $\alpha_{th}$ is the throttle angle, and $u_\alpha \in [0,1]$ is the control input.

Equations (4) and (5) assume that the throttle actuation is neglected. The following relation:

$$\lambda(t) = \frac{1}{\sigma_0}\cdot\frac{\dot{m}_\beta(t)}{\dot{m}_\phi(t)}$$

(6)

describes the ratio of air and fuel in the cylinders with respect to the stoichiometric constant $\sigma_0$[2]. The term $\dot{m}_\phi$ is the fuel mass flow to the cylinder. Assuming that the injectors dynamics and wall-wetting phenomena are negligible and $\lambda$ is approximately constant, we can derive the expression of the fuel flow-mass going into the cylinder as:

$$\dot{m}_\phi = \frac{\dot{m}_\beta(t)}{\sigma_0\lambda},$$

(7)

hence the gas-mixture mass flow aspired in the cylinder is:

$$\dot{m}(t) = \dot{m}_\beta + \dot{m}_\phi = \dot{m}_\beta\left(1 + \frac{1}{\sigma_0\lambda(t)}\right).$$

Finally, the air flow-mass going out from the intake manifold to the cylinders $\dot{m}_\beta$ can be expressed in terms of $\dot{m}(t)$ as follows:

$$\dot{m}_\beta(t) = \frac{\dot{m}(t)}{1 + \frac{1}{\lambda\sigma_0}}$$

(8)

Equation (8) presents the relation between $\dot{m}_\beta$ and $\omega_e$, considering the engine as a volumetric pump, the gas-mixture mass flow aspired in the cylinders is given by:

$$\dot{m}(t) = \frac{p_m(t)}{R\theta_m}\cdot\lambda_l(\omega_e(t),p_m(t))\cdot V_d\cdot\frac{\omega_e(t)}{4\pi}$$

(9)

where $\lambda_l$ is the volumetric efficiency[3]. The volumetric efficiency $\lambda_l$ can be approximated as:

$$\lambda_l(\omega_e,p_m) = \lambda_{l\omega}(\omega_e)\cdot\lambda_{lp}(p_m)$$

(10)

where

$$\lambda_{l\omega}(\omega_e) = \gamma_0 + \gamma_1\omega_e + \gamma_2\omega_e^2$$

(11)

$$\lambda_{lp}(p_m) = \frac{V_c + V_d}{V_d} - \frac{V_c}{V_d}\left(\frac{p_{out}}{p_m}\right)^{\frac{1}{\kappa}}$$

(12)

In Equation (12), $V_c$ is the compression volume, $V_d$ is the volume displacement, $\kappa$ is the ratio of the specific heat ($\kappa \approx 1.4$), and $p_{out}$ is the pressure at the engine's exhaust side. Now we know how $\omega_e$ is related to (1), and to show how $p_m$ is related to (2) we introduce the break mean effective pressure $p_{me}$ which represents the required pressure on the piston during one full expansion stroke to complete an engine cycle[4] and it is given by:

$$p_{me} = \frac{T_e\cdot 4\pi}{V_d}.$$

(13)

The fuel mean effective pressure:

$$p_{m\phi} = \frac{m_\phi\cdot H_l}{V_d},$$

(14)

corresponds to break mean effective pressure generated by an engine that converts all the fuel thermal energy into mechanical energy, for the amount of fuel mass $m_\phi$ burnt per engine cycle. $H_l$ is the specific energy of the fuel. The mass of fuel $m_\phi$ is related to the fuel mass flow by the following equation:

$$\dot{m}_\phi(t) = m_\phi(t)\frac{w_e(t)}{4\pi}.$$

(15)

This equation also represents the relationship that links the mechanical dynamics with the air pressure in the intake manifold.

To exploit this relationship, we start considering the effective efficiency which is:

$$\eta_e = \frac{p_{me}}{p_{m\phi}} = \frac{T_e\cdot 4\pi}{m_\phi\cdot H_f}$$

(16)

and the goal is to find an expression for $T_e$. Hence then:

$$p_{me} = \eta_e(....)\cdot p_{m\phi}$$

(17)

$\eta_e(...)$ can be obtained in different ways, a possible approximation uses the indicated mean pressure:

$$p_{me} \approx p_{mi}(\omega_e) - (p_{m0f}(\omega_e) + p_{m0g}(\omega_e))$$

(18)

where $p_{mi}$ is the indicated mean pressure which is given by:

$$p_{mi} = \frac{w_i}{V_d} = \eta_i\frac{m_\phi\cdot H_f}{V_d}$$

(19)

where $w_i$ is the indicated work[5]. The associated indicated thermodynamic efficiency can be approximated by using the *Willians* approximation as: $\eta_i(\omega_e) \approx \eta_0 + \eta_1\omega_e$. The terms $p_{m0f}$ and $p_{m0g}$ represent the loss due the friction and gas exchange.

---

[1] the temperature of the air flow in input is approximately the same of the air flow in output

[2] represents the optimal air/fuel ratio which generates the maximum heat possible, i.e. for gasoline is 14.66.

[3] describes how far the engine differs from a perfect volumetric device

[4] for a four-stroke engine it corresponds to two engine revolutions.

[5] mechanical energy transferred to the piston during one cycle, where friction is not considered.

From (13), (17), and (18)

$$
\begin{aligned}
T_e(t) &= p_{me}(t) \cdot \frac{V_d}{4\pi} \\
&= (\eta_i(\omega_e) p_{m_\phi} - p_{m0f}(\omega_e) - p_{m0g}(\omega_e)) \frac{V_d}{4\pi} \quad (20)
\end{aligned}
$$

From (16) we find the expression of $p_{m_\phi}$, hence:

$$
T_e(t) = ((\eta_0 + \eta_1 \omega_e) \frac{H_f \cdot m_\phi(t)}{V_d} - p_{m0f}(\omega_e) - p_{m0g}(\omega_e)) \frac{V_d}{4\pi} \quad (21)
$$

By using (15), we can express $m_\phi$ as function of $\dot{m}_\phi$ which can be derived from (7) as:

$$
m_\phi(t) = \frac{\dot{m}_\beta \cdot 4\pi}{\alpha \cdot \omega_e}{}^6 \quad (22)
$$

where $\alpha = \lambda \cdot \sigma_0$.

| param | value | units | param | value | units |
|---|---|---|---|---|---|
| $R$ | 287 | [J/KgK] | $\gamma 1$ | 3.42e-3 | [s] |
| $\theta_a$ | 298 | [K] | $\gamma 2$ | -7.7e-6 | [$s^2$] |
| $\theta_m$ | 340 | [K] | $\eta 0$ | 0.16 | [J/Kg] |
| $\alpha_{th0}$ | 7.9 | [deg] | $\eta 1$ | 2.21e-3 | [Js/Kg] |
| $d_{th}$ | 58.7e-3 | [m] | $\beta 0$ | 15.6 | [Nm] |
| $A_{th,leak}$ | 5.6e-6 | [$m^2$] | $\beta 2$ | 0.175e-3 | [$Nms^2$] |
| $Vd$ | 2.77e-3 | [$m^3$] | $\theta_e$ | 0.2 | [$kg/m^2$] |
| $Vc$ | 0.277e-3 | [$m^3$] | $H_f$ | 45.8e6 | [-] |
| $p_a$ | 1e5 | [Pa] | $\kappa$ | 1.35 | [-] |
| $p_{out}$ | 1e5 | [Pa] | $\alpha$ | 14.70 | [-] |
| $\gamma 0$ | 0.45 | [-] | | | |

TABLE I: Values of the parameters used in the model

Fig. 1 shows the nonlinear model we consider to simulate and control the SI IC engine, where equation (1) has been rewritten by using (3) and (8), and equation (2) with (21) and (22). Table I shows the parameters we use in our simulations [9].

### B. State Space Representation and Controller Design

In this section we present the controller design for our SI engine described by (23) and (24). In this model we can identify $p_m$ and $w_e$ as state variables:

$$
z \triangleq \begin{bmatrix} p_m \\ w_e \end{bmatrix}, \quad (25)
$$

where $h(t) = A_\alpha(t)^7$ as the input, and $w_e$ is the output. The cruise control problem aims to keep a certain constant speed of the car which means that the SI engine is working around a pre-determined equilibrium point $\bar{z} = [\bar{p}_m, \bar{w}_e]^T$ and $\bar{h} = \bar{A}_\alpha$.

A controller for this task can be designed considering the linearized dynamics (23) and (24) around $\bar{x}$. Considering a

---

constant $T_l{}^8$, (23) and (24) can be written as $\dot{z} = f(z, h)$, where $f : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$. From the linearization:

$$
A = \left. \frac{\partial f}{\partial z} \right|_{z=\bar{z}\ h=\bar{h}}, \quad B = \left. \frac{\partial f}{\partial h} \right|_{z=\bar{z}\ h=\bar{h}} \quad (26)
$$

and defining $x \triangleq z - \bar{z}$ and $u \triangleq h - \bar{h}$, the system we consider for design the controller is:

$$
\dot{x} = Ax + Bu + w \quad (27)
$$

where $w \sim \mathcal{N}(0, Q)$ represents the process noise which is zero-mean Gaussian with known covariance matrix $Q$. Since the output is $w_e$, then the output equation of the linearized model is $y = Cx + v$ with $C = [0\ 1]$ and $v \sim \mathcal{N}(0, R)$ is the zero-mean Gaussian measurement noise with variance $R$. To design the controller we consider the discrete version of (27) with sampling time $T_s$. Hence the considered system is given by:

$$
\begin{aligned}
x_{k+1} &= A_d x_k + B_d u_k + w_k \quad (28) \\
y_k &= C x_k + v_k \quad (29)
\end{aligned}
$$

with $w_k \sim \mathcal{N}(0, Q_d)$, and $v_k \sim \mathcal{N}(0, R_d)$. We defined $A_d$, $B_d$, $Q_d$, and $R_d$ as in [22]. To stabilize the dynamics (28), we consider a Linear Quadratic Gaussian (LQG) controller. This is a state feedback controller $u_k = Lx_k$ that minimizes the following cost function:

$$
J = \lim_{N \to \infty} \mathbb{E} \frac{1}{N} \left[ \sum_{k=0}^{N-1} (x_k^\top W x_k + u_k^\top U u_k) \right], \quad (30)
$$

where $W$ and $U$ are semi-positive definite matrices. The actual control is computed as:

$$
u_k = L \hat{x}_{k|k} \quad (31)
$$

where $\hat{x}_{k|k}$ is the Kalman state estimation at time $k$

$$
\hat{x}_{k|k} = \hat{x}_{k|k-1} + K \left( y_k - C \hat{x}_{k|k-1} \right). \quad (32)
$$

The one-step prediction $\hat{x}_{k|k-1}$ is given by:

$$
\hat{x}_{k|k-1} = A_d \hat{x}_{k-1|k-1} + B_d u_{k-1}. \quad (33)
$$

The constant gain $K$ is the Kalman gain considered at steady state. For more details about (31) and (32), the reader is invite to see [22]. The associated observer-based state feedback control scheme is reported in Fig. 3.

### C. The Controller Area Network

The Controller Area Network (CAN) is a vehicle bus standard designed to allow the nodes of the network to exchange data without requiring a host computer [1]. CAN is one of the most deployed networking protocol for internal vehicular communications due to its high resilience to electromagnetic interference and its cheap implementation. Microcontrollers on the same CAN segment exchange data between themselves

---

[6]for idle control gas-mixing transportation delays have to be taken into account, $\dot{m}_\phi(t - \delta)$ and $\dot{m}_\beta(t - \delta)$.

[7]To do not overload the representation we consider directly $A_\alpha(t)$ as input. To consider $u_\alpha$ the throttle angle, we need to consider the equations (4) and (5).

[8]for a case of replay attack, a time-varying $T_l$ makes the attack not stealthy, here we consider a scenario where $T_l$ is constant and possible small changes can be modelled as random noise.

$$\frac{dp_m(t)}{dt} = \frac{R\theta_m}{V_d}\left(A_\alpha(t)\frac{p_a}{\sqrt{R\theta_a}}\frac{1}{\sqrt{(2)}} - \left(\frac{p_m(t)}{R\theta_m}\left(\gamma_0 + \gamma_1\omega_e(t) + \gamma_2\omega_e^2(t)\right)\right.\right.$$
$$\left.\left.\left(\frac{V_c + V_d}{V_d} - \frac{V_c}{V_d}\left(\frac{p_{out}}{p_m}\right)^{\frac{1}{\kappa}}\right)\frac{V_d\omega_e(t)}{4\pi}\frac{\alpha}{\alpha+1}\right)\right) \tag{23}$$

$$\frac{d\omega_e(t)}{dt} = \frac{1}{\theta_e}\left[\left((\eta_0 + \eta_1\omega_e(t))\frac{H_f \cdot p_m(t)}{R\theta_m}\left(\gamma_0 + \gamma_1\omega_e(t) + \gamma_2\omega_e^2(t)\right)\left(\frac{V_c + V_d}{V_d} - \frac{V_c}{V_d}\left(\frac{p_{out}}{p_m}\right)^{\frac{1}{\kappa}}\right)\right.\right.$$
$$\left.\left.\cdot\frac{V_d}{\alpha+1} - \left(\beta_0 + \beta_2\omega_e^2(t) + (p_{out} - p_m(t))\right)\frac{V_d}{4\pi}\right) - T_l(t)\right] \tag{24}$$

Fig. 1: Nonlinear model of an SI-engine for cruise-control problem.

using the CAN data frame, one of the 4 types of frames defined by the CAN standard. The CAN data frame is composed of two main fields, namely the *identifier (ID)*, and the *payload (data)*. The *ID* is used to distinguish among different types of CAN data frame. Data frames characterized by a given ID are produced by only one microcontroller, while receiver microcontrollers use the value of the ID to select data frames that are relevant for their functioning. The CAN standard defines two types of data frames: the *standard* format, whose ID field is 11-bits long, and the *extended* format, whose ID field is encoded using 29-bits. The extra bits of the extended format are separated from the bits composing the standard format for backward compatibility. Figure 2 shows an example of a generic CAN data frame in the *extended* format. The *data* field encapsulates the information that the sender microcontroller transmits to other microcontrollers on the network. The data field has a variable size (from 1 to 8 bytes) and usually packs several different signals. The CAN standard leaves complete freedom to the car manufacturers about the structure, number, encoding, and semantic of these signals. Hence, without having access to the formal specifications of the CAN network for a particular vehicle model, the signals encoded in the data field can only be interpreted as an opaque binary blob.
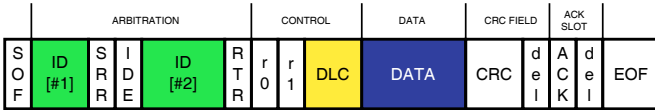


Fig. 2: CAN data frame in the extended format

For the tests presented in this paper, we designed a CAN communication network between the IC engine model the speed controller. In particular, we identified a minimum of two messages required to enable communication between the IC engine model and the speed controller. The first message, called *ES (engine speed)* is sent from the ECU attached to the *engine output sensor* and required as input by the *controller*, while the second message is called *TR (throttle request)* and is sent from the *controller* to the input of the *engine model*. The *ES* message contains the value of the engine rotational speed, while the *TR* message contains the value of the throttle estimated by the controller to keep the rotational speed constant. The detailed description of the two CAN messages is provided in Table II.

| Message | ID [hex] | cycle time [ms] | encoded signal | DLC [bits] |
|---------|----------|-----------------|----------------|------------|
| ES | 0x10 | 10 | throttle opening request [0, 1] | 32 |
| TR | 0x15 | 10 | engine rotational speed [RPM] | 32 |

TABLE II: CAN bus specifications

Figure 3 presents the final architecture of the system, showing how the different subsystems are connected with each other: a black line is used to denote direct connection between the subsystems, while the orange line is used to highlight subsystems communicating via CAN.
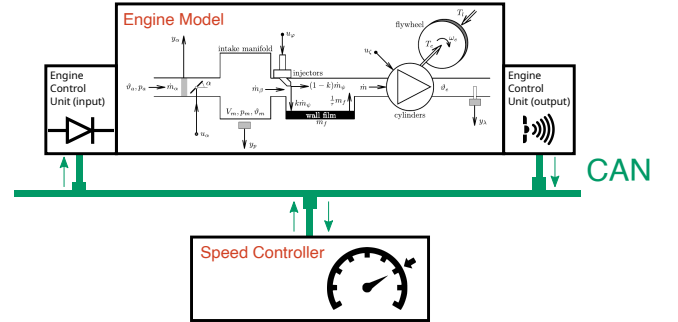


Fig. 3: Final design of the IC engine model with the controller and CAN communication

## IV. THREAT MODEL

In this section we present the threat model considered for testing our model. The threat model considered in this paper is focused on attacks targeting the output of the engine model, with the final goal to modify its output.

Modern literature already showcased different threat models for both CPS security and CAN communications, demonstrating the vulnerabilities of modern control systems and networking protocols. The consequences of these vulnerabilities have

been exploited by security researchers from both academia and industries to perform different attacks, from simple *Denial-of-Services* [8], [21], designed to test the systems against a disruption of the communication, to more advanced *impersonation attacks* [23], in which the attacker is able to impersonate a target ECUs by replacing legit messages with maliciously forged ones. However, all the presented threat models are developed by focusing only on a specific aspect of the model, being either the control system itself [24], [25] or the adopted communication protocol [26], [27].

The threat model considered in this paper is based on both research areas, hence representing a hybrid approach composed of control system theory and CAN communication. To the best of our knowledge, this is the first time a hybrid approach is used for the definition of a threat model for automotive applications. The threat model considered in this paper is composed of three different attack scenarios, each one designed to modify the output of the system by means of different attack vectors. The detailed description of each attack scenario is presented in the following.

### A. Fuzzing attack

The fuzzing attack considered in our threat model requires an attacker with the ability to provide custom input to the control system, resulting in the modification of the speed of the engine. This attack supposes that the attacker is able to inject a custom control input to the system at anytime. The attack is composed of two stages: in the first stage the attacker learns the legit values of the input of the control system, while in the second stage a desired sequence of control input is provided to the system. The fuzzing attack is a type of attack already explored on both control systems [28] and CAN communications [29], [30]. However, since the input of the system can be modified at sensor level or by tampering the content of its relative CAN message, we remark that the consequences of the fuzzing attack scenario can be analyzed by targeting either the model's sensors or the CAN communication.

### B. Replay attack

The replay attack is an advanced version of the *fuzzing attack* scenario, in which it is necessary that the attacker is able to provide custom input to the control system while simultaneously change its output by replaying the corresponding output of the provided input. This attack scenario is composed of two stages: in the first stage the attacker records a sufficient number of sensor readings (both input and output) without providing any input to the system, hence with the system running in normal conditions; while in the second stage the attacker provides a desired sequence of control input while replaying the previously recorded outputs. By conducting a replay attack to the powertrain system, the attacker is able to modify the final speed of the vehicle without being noticed, by injecting a control input to the system and masquerading its consequences to the controller by replaying previously

recorded output values. This attack scenario has already explored on both research areas [31], [32] hence we remark that its consequences on our model can be analyzed by simulating the attack on one of the two scenarios.

### C. Injection attack

In the injection attack scenario, the attacker has no direct access to the sensors attached to the ECUs but is able to read and send CAN messages, either via a physical connection or by exploiting a vulnerability of a target ECU. The injection attack is an attack targeting the CAN communication between the system microcontrollers, in which the attacker is able to send messages containing an arbitrary value of the system control input to modify its output. The injection attack is designed in two stages: in the first stage the attacker observes the CAN communication of the target vehicle to identify the messages carrying the input values of the control system and to learn their cycle time; while in the second stage the attacker injects CAN messages with the desired control input encoded in their data field with a faster cycle time than the original ones, interleaving malicious messages with the valid ones. As opposed to the previous two attack scenarios, the injection attack can only be performed at CAN communication level [23].

## V. ATTACK CONSEQUENCES

In this section we investigate the consequences of the attacks composing the threat model presented in Section IV on the powertrain model presented in Section III. The attacks are replicated on our model after $10$ seconds of normal system simulation for a duration of $2$ seconds.

### A. Fuzzing attack

The consequences of the fuzzing attack on our model are investigated by performing the attack at CAN level. The attack is simulated as follows. In the first phase of the attack we identified a sequence of input values corresponding to $2$ seconds of system inputs. In the second phase we changed the real value of the sensor reading encoded in its relative CAN message by increasing the values of a fixed amount. The attack analyzed in this section targets the input of the engine model (the throttle request), and the value encoded in the CAN messages is increased by $1e-6$. Figure 4 shows the consequences of this attack on our system, comparing the input (throttle request, blue line) with the output (engine speed, green line). The left $y$-axis shows the values of the input, the right $y$-axis shows the values of the output, while the $x$-axis shows the time of the simulated system. We denote the start of the attack with a red vertical line.

From the analysis of the results presented in Figure 4 we notice that after the start of the attack the output of the system rapidly increased, from its stationary value of $4200$rpm to a peak of $4224$rpm. Moreover, despite the input of the system is fixed at $7.54e-5$, the action of the cruise controller deployed on our model stabilizes the input of the system to approximately $7.462e-5$, hence slightly decreasing the malicious input.
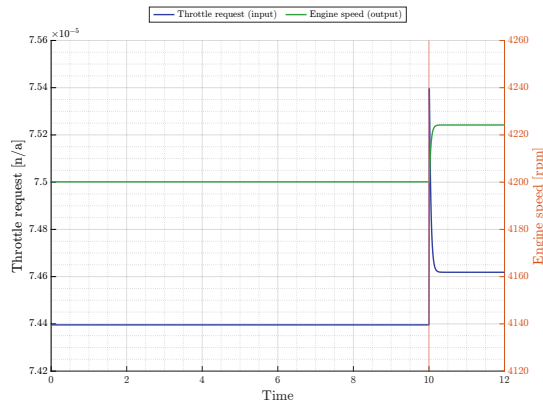
Fig. 4: Consequences of the fuzzing attack on the system input

## B. Replay attack

The consequences of the replay attack on our model are investigated by performing the attack at CAN level. For this attack scenario, we also remark that the same results are achieved by performing the same attack at sensor level. The attack is conducted as follows. In the first phase of the attack we recorded a sequence of input and output values corresponding to 2 seconds. The second phase of the attack is conducted as in the *fuzzing attack* scenario, despite in this attack scenario the content of the CAN message carrying the reading's of the output sensor is overwritten with the output value corresponding to the given input. Figure 5 shows the consequences of this attack on our system, comparing the input (throttle request, blue line) with the output (engine speed, green line). The left $y$-axis shows the values of the input, the right $y$-axis shows the real output of the system (and not the replayed values), while the $x$-axis shows the time of the simulated system. The start of the attack is represented by a red vertical line.
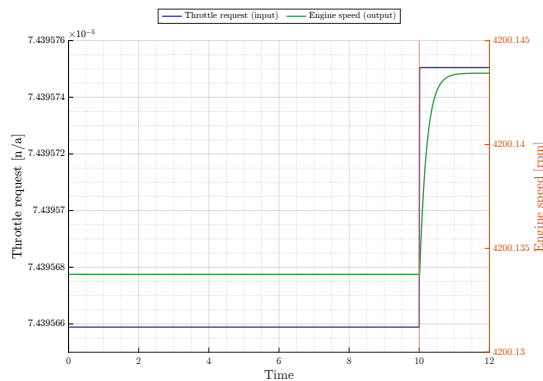


Fig. 5: Consequences of the replay attack on the system

Figure 5 shows that after the start of the replay attack there is an increment of the actual speed of the engine. Moreover, by comparing the evolution of the input of the system with the results presented in Figure 4, it is possible to notice that by overwriting the output of the system before its relative CAN

message is sent to the controller it is possible to prevent the controller to operate on the input.

## C. Injection attack

The consequences of the injection attack scenario are investigated by performing the attack at CAN level. We remark that this attack scenario is limited to the CAN communication bus, hence it is not possible to obtain the same results by performing the attack at sensor level. The injection attack is performed by injecting throttle request CAN message with a higher value of the throttle input to increase the speed of the engine. The attack is conducted as follows. In the first phase the legit values of the input are analyzed to identify the normal value used by the controller. In the second phase, messages carrying a higher value of the input of the system are injected with a frequency 10 times higher than the legit message, thus injecting 10 malicious messages between two legit ones. The input value of the injected message is $1e-6$ higher than the value sent by the original messages.

Figure 6 shows the consequences of this attack on our system, comparing the input (throttle request, blue line) with the output (engine speed, green line). The left $y$-axis shows the values of the input, the right $y$-axis shows the output of the system, while the $x$-axis shows the time of the simulated system. The start of the attack is represented by a red vertical line. The internal plot of Figure 6 is a detailed example of the system input and output centered at time $t = 11$ seconds. In this internal Figure it is possible to notice better the difference of the between the legit values of the input (blue dot) and output (green dot) sensors, the injected system input (blue crosses) and its relative output (green crosses).
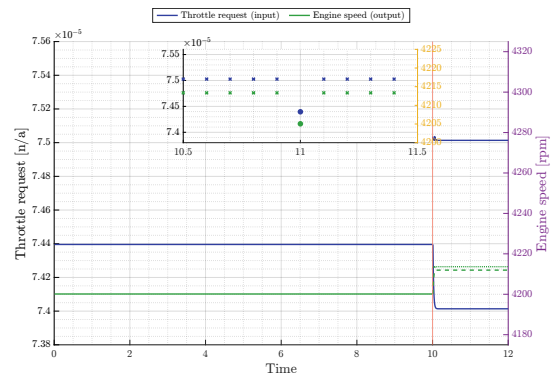


Fig. 6: Consequences of the injection attack on the system

The analysis of the results presented in Figure 6 shows an interesting pattern caused by the start of the attack. In particular, the output of the system goes from a minimum values of 4120rpm (which to the legit value) to values of 4215rpm (which corresponds to the injected input).

## VI. CONCLUSIONS

In this paper we analyzed the consequences of cyber-attacks to the powertrain model of a generic vehicle. The powertrain model is built by considering a generic internal combustion

engine model, composed of different sub-models, a speed controller, and a communication network based on the CAN protocol connecting the engine model with the controller. We consider a threat model composed of different attack scenarios, whose final goal is to modify the output of the engine model, representing the rotational speed of the engine. We discussed 3 different attacks to the system, of which 2 can be deployed at both sensor and CAN level, while the other attack can be only executed at CAN communication level. We experimentally analyzed the consequences of these attacks on our model through simulation of the different attacks, demonstrating that it is possible to modify the output of the model by introducing little deviations in the content of the CAN messages or in the sensor's readings. Compared to the current state-of-the-art, the work presented in this paper uses a cruise controller developed to operate in conjunction with the model describing the actual components of a generic internal combustion engine, demonstrates the consequences of cyber-attacks targeting the powertrain system, and uses the CAN communication protocol to enable communication between the engine and the controller.

Future work is focused on the analysis of state-of-the-art defences against the considered threat model to build a framework that exploits the two scenarios to overcome each other limitations.

## REFERENCES

[1] Bosch. (1991) Can specification version 2.0.
[2] C. Miller and C. Valasek. (2014) Adventures in automotive networks and control units.
[3] Keen Security Lab of Tencent. (2016) Car hacking research: Remote attack tesla motors. [Online]. Available: http://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/
[4] C. Miller and C. Valasek. (2015) Remote exploitation of an unaltered passenger vehicle. White paper of Blackhat US conference. [Online]. Available: http://illmatics.com/Remote\%20Car\%20Hacking.pdf
[5] M. Marchetti and D. Stabili, "Anomaly detection of can bus messages through analysis of id sequences," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 1577–1583.
[6] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 787–800. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243751
[7] D. Stabili and M. Marchetti, "Detection of missing can messages through inter-arrival time analysis," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–7.
[8] Q. Tianxiang, H. Defeng, L. Liangye, and S. Xiulan, "Adaptive cruise control of vehicles subject to denial-of-service," in *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2017, pp. 382–386.
[9] L. Guzzella and C. Onder, *Introduction to modeling and control of internal combustion engine systems*. Springer Science & Business Media, 2009.
[10] A. Kırlı, Y. Chen, C. E. Okwudire, and A. G. Ulsoy, "Torque-vectoring-based backup steering strategy for steer-by-wire autonomous vehicles with vehicle stability control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7319–7328, 2019.
[11] A. Patel and M. Ferdowsi, "Current sensing for automotive electronics—a survey," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 8, pp. 4108–4119, 2009.
[12] M. Ruff, "Evolution of local interconnect network (lin) solutions," in *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, vol. 5, 2003, pp. 3382–3389 Vol.5.
[13] AUTOSAR. (2019) Specification of Secure Onboard Communication AUTOSAR CP R19-11. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards/classic/19-11/AUTOSAR_SWS_SecureOnboardCommunication.pdf
[14] G. Bella, P. Biondi, G. Costantino, and I. Matteucci, "CINNAMON: A module for autosar secure onboard communication," in *2020 16th European Dependable Computing Conference (EDCC)*, 2020, pp. 103–110.
[15] F. Pollicino, D. Stabili, G. Bella, and M. Marchetti, "Sixpack: Abusing abs to avoid misbehavior detection in vanets," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–6.
[16] F. Pollicino, D. Stabili, L. Ferretti, and M. Marchetti, "Hardware limitations to secure c-its: Experimental evaluation and solutions," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12 946–12 959, 2021.
[17] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Snachám, and S. Savage, "Experimental security analysis of a modern automobile," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 447–462, 2010.
[18] S. Chakraborty, M. A. Al Faruque, W. Chang, D. Goswami, M. Wolf, and Q. Zhu, "Automotive cyber–physical systems: A tutorial introduction," *IEEE Design & Test*, vol. 33, no. 4, pp. 92–108, 2016.
[19] P. Griffioen, S. Weerakkody, B. Sinopoli, O. Ozel, and Y. Mo, "A tutorial on detecting security attacks on cyber-physical systems," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 979–984.
[20] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2009, pp. 911–918.
[21] L. Zhang, Y. Chen, and M. Li, "Resilient predictive control for cyber-physical systems under denial-of-service attacks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2021.
[22] B. Yaghooti, R. Romagnoli, and B. Sinopoli, "Physical watermarking for replay attack detection in continuous-time systems," *arXiv preprint arXiv:2103.00790*, 2021.
[23] N. Nowdehi, W. Aoudi, M. Almgren, and T. Olovsson, "CASAD: CAN-aware stealthy-attack detection for in-vehicle networks," 2019.
[24] C. Schmittner, Z. Ma, E. Schoitsch, and T. Gruber, "A case study of fmvea and chassis as safety and security co-analysis method for automotive cyber-physical systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, ser. CPSS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 69–80. [Online]. Available: https://doi.org/10.1145/2732198.2732204
[25] A.-M. Jamil, S. Khan, J. K. Lee, and L. ben Othmane, "Towards automated threat modeling of cyber-physical systems," 2021.
[26] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028073
[27] S. Abbott-McCune and L. A. Shay, "Techniques in hacking and simulating a modem automotive controller area network," in *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, 2016, pp. 1–7.
[28] Y. Chen, C. M. Poskitt, J. Sun, S. Adepu, and F. Zhang, "Learning-guided network fuzzing for testing cyber-physical system defences," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019, pp. 962–973.
[29] H. Lee, K. Choi, K. Chung, J. Kim, and K. Yim, "Fuzzing CAN packets into automobiles," in *2015 IEEE 29th Int'l Conf. on Advanced Information Networking and Applications*, March 2015.
[30] F. Martinelli, F. Mercaldo, V. Nardone, and A. Santone, "Car hacking identification through fuzzy logic algorithms," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–7.
[31] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2009, pp. 911–918.
[32] B. Groza and P.-S. Murvay, "Efficient intrusion detection with bloom filtering in controller area networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1037–1051, 2019.