Janus: A Trusted Execution Environment Approach for Attack Detection in Industrial Robot Controllers

Stefano Longari, Jacopo Jannone, Mario Polino, Michele Carminati, *Member IEEE,* Andrea Zanchettin, *Member IEEE,* Mara Tanelli, *Senior Member IEEE,* and Stefano Zanero, *Senior Member IEEE.*

Abstract—In the last few decades, technological progress has led to a spike in the adoption of robots by the manufacturing industry. With the new "Industry 4.0" paradigm, companies strive to automate their production processes by interconnecting and integrating different industrial systems. The resulting increase in complexity contributes to a larger attack surface and paves the way for novel attacks. In the context of cyber-physical systems, consequences include economic and physical damage, as well as harm to human workers. In this paper, we present Janus, a novel monitoring mechanism for industrial robot controllers that exploits the trusted execution environment (TEE) to guarantee the integrity of the attack detection algorithm even in case the controller's software is compromised, while not requiring external hardware for its detection process. In particular, we use the state observers strategy for detecting low-level controller (LLC) attacks.

We assess our approach by testing it against various attacks, identifying those that are simpler to detect and pinpointing the more elusive ones, which are mostly detected nonetheless. Finally, we demonstrate that our approach does not add significant computation overheads.

Index Terms—industrial robots, TrustZone, trusted execution environment, attack detection, cyber-physical systems.

1 Introduction

Industrial robots play a crucial role in the modern industrial world, as they can increase manufacturing efficiency, reduce cost, and increase output quality. The global number of units in operation has nearly tripled in the last ten years, and growth shows no signs of stopping [1]. Most industrial robots are capable of fast, powerful, and precise movements, which compose safety hazards to factory workers. Traditionally, risks have been mitigated by letting industrial robots operate in cages, isolated from human operators by physical barriers.

However, technical advances are leading to increased use of collaborative robots, or cobots, which operate in a shared workspace alongside human workers [2, 3]. These robots heavily rely on software features to ensure safety [4, 5]; hence, an attacker that gains control over the robot software may bypass these safety measures, allowing the robot to perform dangerous and unexpected movements. Recent digital transformation has resulted in "smart factories" adopting increasingly complex and interconnected machines. Benefits of connected robots include the ability to program, control, and continuously monitor them at a remote location, improving efficiency and minimizing production downtime. However, these innovations come at the cost of an increased attack surface that could enable attackers to compromise devices in ways that were not feasible before. What is more, many machines that are currently part of interconnected factory networks - or worse, exposed to the Internet - were initially designed to work in isolation and lack proper security measures or mitigation strategies to existing vulnerabilities [6]. The most sensitive component of an industrial robot is the controller. Nevertheless, almost no research from the current state of the art deals with solutions against controller attacks that aim to preserve the flexibility that characterizes industrial robots.

Existing works detect such attacks through host-based intrusion detection systems, without tackling the issue of deploying the IDS. However, it has been demonstrated that an attacker can obtain code execution on the robot, compromising its functionalities [6, 7, 8]. Given that TEE-based technologies are spreading in the context of industrial microcontrollers and are considered an effective hardware-based security solution in modern microprocessors, we deem it necessary, for the security community to study and understand all the novel ways in which they can be applied as a countermeasure.

To bridge this gap, we present Janus, a novel approach to the security of robot controllers that makes use of the trusted execution environment (TEE) implemented in modern microcontrollers to guarantee the integrity of the detection system even in the case its software is compromised without requiring additional hardware. For the detection of attacks in the LLC, we adopt a widely used strategy in robotics for collision detection [9], based on *state observers*, and exploit it as an attack detection mechanism for industrial robots. By leveraging TrustZone, differently from existing approaches, we ensure the security of our attack detection algorithm so that its detection capabilities remain intact even in case the

All authors are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy. E-mail: {stefano.longari, mario.polino, michele.carminati, mara.tanelli, stefano.zanero}@polimi.it. jacopo.jannone@mail.polimi.it

Manuscript received October 30th, 202 2; revised X.

LLC software is completely compromised, while eliminating the need for implementing the system on a separate machine.

We implement our solution using Matlab/Simulink and test it against several different attacks. Results show that the proposed system can effectively detect attacks that threaten the surrounding environment, human operators, and the robot itself. Attacks that slowly diverge from the expected trajectory are the most difficult to detect, representing a limitation in high-precision applications where tolerances on the robot trajectory are particularly tight.

Overall, this work brings the following contributions:

- A novel approach that leverages the security capabilities of the trusted execution environment (TEE) found in modern MCUs to ensure the integrity of a LLC attack detection system for industrial robots, with the system being implemented on the same hardware that it aims to protect.
- A novel application of the state observer strategy for detecting attacks against the low-level components of industrial robot controllers.
- A quantitative experimental evaluation of Janus in terms of accuracy, time to detection, and maximum trajectory deviation on a simulated scenario and on a physical, TrustZone-equipped development board.

The rest of the paper is structured as follows: in Section 2 we introduce the main concepts related to industrial robots and we briefly present the Arm TrustZone technology. In Section 3 we compare several proposals from the current state of the art, highlighting their strengths and limitations. Section 4 illustrates the threat and attacker models considered for our solution. We describe our approach and the architecture of our system in Section 5. Then, in Section 6 we detail the experimental validation of our system and discuss the results. Finally, we present our conclusions in Section 7.

2 Background and Motivation

2.1 Industrial Robots

Industrial robots are mechanical, multi-axis arms operating under computer control [10, 11]. Their main mechanical component is the *manipulator*, which consists of multiple segments (*links*) connected to one another by motor-powered *joints*. The manipulator allows moving objects with several degrees-of-freedom (degrees of freedom (DOF)), depending on the number of joints. From a system's dynamics perspective, each joint *state* can be described by its position, velocity, and acceleration, which are constantly measured by *sensors* reporting data to the controller. An *end effector* (consisting of grippers or other specialized tools) is connected to the manipulator's end effector to interact with the environment. Industrial robots can be programmed either using a *teach pendant* or offline, with a proprietary programming language that allows users specifying robot movements using Cartesian coordinates. The robot *controller* performs the computational and power functions required to operate a robot [12]. It can be split into a high-level and a low-level controller (HLC and LLC, respectively). The former usually consists of an x86 processor-based computer running a real-time operating system. It works in an open-loop fashion, and it computes each joint's *target state* at each time instant, following programmed instructions. The LLC is microcontroller-based, and it implements a closed-loop control system that finely adjusts motors torques to constantly keep each joint at its target state.

2.2 Arm TrustZone

Our Proof of Concept (PoC) is based on ARM TrustZone, which is a hardware security extension of the Arm processor architecture that leverages the principle of security by isolation. The processor is partitioned into a *secure world* and a *normal world*. Strong isolation between the two worlds is enforced at the hardware level so that software running in the normal world is unable to access resources belonging to the secure world. The partition is done by mapping the memory addresses such that they can be accessed from one or the other state. Since microcontrollers' peripherals are mapped as memory addresses, it is possible to deny access to their data from the non-secure state if needed. By doing so, TrustZone enables developers to implement critical operations in a secure environment without having to trust existing software, such as the operating system.

After having become available for application processors (Cortex-A) for several years, TrustZone has been added to Cortex-M microcontrollers as well, optimized for low-power and real-time systems. Memory addresses and interrupts can be configured to belong to either the normal or the secure world. Non-secure interrupts can be deprioritized or disabled entirely during execution in the secure world so that software running in the normal world cannot disrupt security-critical processes [13].

While multiple vulnerabilities have been found for TrustZone [13, 14], to the best of the authors' knowledge all such vulnerabilities are related to existing bugs in the TEE kernel and TEE drivers implementation of some providers, or they are vulnerabilities in systems built on TrustZone, rather than being inherent issues in the TrustZone technology itself.

2.3 Motivation

With this paper, we aim to address the scarcity of solutions to ensure the security of attack detection algorithms in industrial robot controllers. Our goal is to design a monitoring system that leverages a Trusted Execution Environment to preserve the integrity of the detection algorithm, ensuring that its effectiveness remains uncompromised. Our approach needs to reliably determine if the robot behaves as expected or if its movements differ from those required to carry out the current task. We design our system to be flexible enough not to need a reconfiguration when the robot task changes. To the best of our knowledge, this work is the first to implement such system on industrial robot controllers.

3 Related Works

In the context of physics-based attack detection techniques against cyber-physical systems (CPSs), most of the current literature focuses on attacks that target sensors, actuators, and physical signals on the network [15, 16, 17]. Several works deal with the problem of state estimation when a subset of sensors is under attack, and algorithms exist to efficiently tell malicious and legitimate sensors apart [18, 19, 20, 21]. Some solutions propose physical watermarking and challenge-response mechanisms to authenticate sensor and actuator signals [22, 23, 24], while others concentrate upon ensuring the stability of the physical system subject to attack [24, 25].

Few studies deal with the detection of attacks against the controller, and most of them are only applicable to industrial robots with the assumption that the robot performs a single repetitive task [26, 27, 28]. Since they require a reconfiguration or recalibration each time the robot task changes, their versatility is significantly limited.

Recently, Pu et al. [29] studied the correlation between the movements of an industrial robot and its power consumption, and trained an artificial neural network to detect anomalies caused by attacks. To the best of our knowledge, their approach is the only one capable of detecting attacks against the robot controller independently of the robot task. Nevertheless, the system was only tested against anomalies at a macroscopic level, and it offers little or no explainability of the detected anomalies.

Abdi et al. [30] presented a restart-based approach for limiting the impact of attacks against the controller of CPSs, leveraging the fact that physical systems cannot be destabilized instantaneously due to their finite inertia. However, their solution only works for stateless controllers and "stable" physical systems, in the sense that can tolerate short intervals of anomalous behavior without critical consequences.

Hasan and Mohan [31] developed *Contego-TEE*, a framework that combines TrustZone with an invariant-checking mechanism to protect Internet of things (IoT) devices from malicious control commands. Defining a set of invariant conditions at design time is feasible for simple and single-task systems; however, it quickly becomes overly complicated and inflexible as system complexity increases.

4 Threat Model

4.1 Threat Scenarios

The current state of the art [7] identifies four main threats that a compromised industrial robot may pose: physical damage, production sabotage, denial of service, and loss of data confidentiality. In this research, we focus on the first two, which are those that modify the physical behavior of a system and therefore the most critical ones in the context of CPSs.

Physical damage. An attacker may change the controller's behavior, forcing the robot to perform unexpected actions, including movements that violate spatial and speed constraints. Such actions may result in damage to machinery, injury to human operators, as well as harm to the robot itself. This scenario is the most critical one, both because of its impact on safety and because of the economical damage resulting from the long-term downtime a successful attack would cause.

Production sabotage. An attacker may cause small changes to the controller behavior to inject micro-defects into the final product, which may translate into an increased failure rate, shorter lifespan, and lower quality. The impact of this kind of attack becomes clear when considering that the final product may be used in safety-critical fields, such as medical, military, and transportation, potentially resulting in indirect severe or fatal consequences. In these safety-critical fields in-depth quality assurance testing is usually performed to assess the quality of the product before shipping it to the final customer. However, finding the source of the defects introduced by an attacker is extremely challenging and may results in significant economic damages [6].

4.2 Attacker Model

We consider attackers knowledgeable of the logical structure of industrial systems and industrial robots. They know which software and operating systems are commonly used in robot controllers and their security features. They can resort to reverse engineering to learn how to reprogram (or patch) the LLC to alter the robot's behavior in a controlled manner. We distinguish between two main types of attackers according to their system access level.

External attackers. They do not have prior access to factory systems, but they possess advanced network exploitation capabilities. They may be able to take over vulnerable robot controllers exposed to the Internet by means of "industrial routers"[7], used by vendors to provide remote support, monitoring and maintenance services. Alternatively, they might exploit vulnerabilities in other exposed systems, reach the robot controller throughout the factory local area network (LAN), and bypass any authentication mechanism to its application programming interfaces (APIs).

Internal attackers. They are insiders with physical access to the factory, either led by personal motives or manipulated through social engineering techniques. The lack of restrictions on physical interfaces could allow them to use a malicious USB drive to infect the controller with malware. Similarly, they could use any available I/O interface, such as Ethernet, serial, or CAN ports, to interact with it through the network.

4.3 Attack Scope

Our scope includes all software attacks that target the LLC. Attacks that involve physical access are included as long as the physical interaction is limited to the external interfaces of the controller. We trust peripheral devices such as sensors and actuators. In the following, we delineate three notable attack classes.

Alteration of control loop parameters. An attacker may change parameters characterizing the control law under which the robot controller operates. They are usually tuned at design time to ensure the stability of the control loop. An alteration may cause anomalies ranging from very slight variations of the nominal trajectory to system instability and mechanical breakage of the robot.

Alteration of input processing. An attacker may tamper with the portion of controller code that processes and decodes input signals collected from sensors before they are used in the control loop. Such an attack would result in the use of wrong information about the current state of the robot and, therefore, in performing incorrect actions.

Alteration of output processing. Similarly, an attacker may tamper with the portion of controller code that encodes output signals before sending them to the motor drives. In this case, the attacker may directly impact the behavior of actuators after the correct control action has been computed.

We emphasize that attacks affecting the HLC are not included in our scope; thus, we do not contemplate malicious variations of the robot task. Our detection system uses the HLC output as a trusted reference against which sensor signals (also trusted) are verified. Therefore, we assume that the HLC always delivers correct values for the intended robot task. We remark that protecting the HLC is already possible with more traditional techniques, such as trusted computing [32]. Moreover, being external to the control loop, operations performed by the HLC can be replicated in a simulation environment, the output of which can be used as a reference for checking the output of the real system.

5 Janus's Approach

The concept behind Janus is that of implementing an efficient attack detection algorithm directly in the robot controller system, without requiring additional hardware, while defending it from tampering by potential attackers through the use of TEE. For the detection of attacks, we export the control-oriented design of state observers, used in robotics for, e.g., collision detection [9]. By doing so, we focus on detecting the physical effects of the attack on the behavior of the robot, making our approach general and independent from any specific attack vector and from the robot's task. Moreover, our approach does not require external hardware or any changes to the control algorithm. We execute critical steps of attack detection in a secure environment using TrustZone, a security extension of Arm microcontrollers, which defends the integrity of our system. A key part of this process involves identifying and securing the critical control routines and data vital to the detection algorithm, which must be secured to prevent tampering by any potential attacker.

5.1 Attack Detection

A state observer is an algorithm that estimates an unmeasurable internal state of a dynamical system, given some knowledge of its mathematical model and its inputs and outputs. In our approach, the considered mathematical model is made of the Ordinary Differential Equations (ODE) describing the robot's dynamics; inputs are data from the HLC; outputs are data sensors' measurements. One of the advantages of model-based techniques is that no additional hardware is needed, and all required data is already available for the control of the robot.

We base the design of our state observer on the robot *generalized momenta*, following an approach initially proposed by De Luca and Mattone [33] for actuator failure detection and isolation. We consider a generic n-DOF robot manipulator subject to attacks, the dynamic model of which is expressed by, e.g., [11]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau + \tau_K, \tag{1}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ represent each joint angular position, velocity, and acceleration; $\boldsymbol{M}(\boldsymbol{q})$ is the robot inertia matrix;

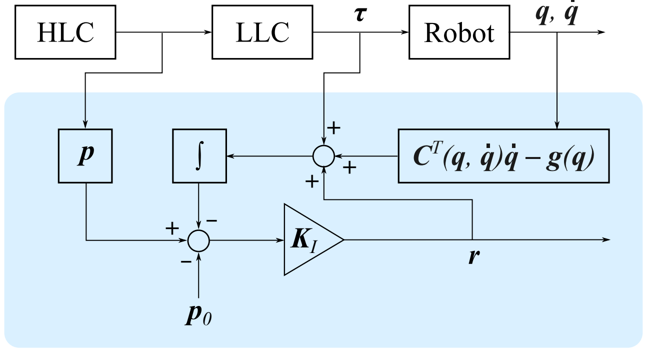


Figure 1: Block diagram representation of the residual generation algorithm.

$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is the Coriolis/centrifugal matrix; $\boldsymbol{g}(\boldsymbol{q})$ is the gravity vector; $\boldsymbol{\tau}$ is the vector of motor torques; $\boldsymbol{\tau}_K$ represents the external torques caused by attacks. Note that also in the model proposed in [33] the attacker's torque is modeled as an additive input, as its aim is that of altering the robot's movement with malicious commands.

The attack detection strategy consists of two steps. The first is the generation of a diagnostic signal, called *residual signal*, which quantifies the gap between the amount of torque absorbed by the motors and the amount of torque that the same motors are estimated to absorb based on the robot model to follow the reference trajectory. More specifically, following [33], we define the *residual vector* $\boldsymbol{r}$ as

$$\boldsymbol{r}(t) = \boldsymbol{K}_I\left[\boldsymbol{p}(t) - \int_0^t \Big(\boldsymbol{\tau} + \boldsymbol{C}^T(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{r}\Big)ds - \boldsymbol{p}(0)\right], \tag{2}$$

where $\boldsymbol{K}_I$ is a gain matrix; $\boldsymbol{p} = \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$ is the vector of generalized momenta of the robot; $\boldsymbol{\tau}$ is obtained from the LLC; $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are obtained from robot sensors. Figure 1 represents the residual computation in the form of a block diagram. It can be shown [33] that

$$\dot{\boldsymbol{r}} = \boldsymbol{K}_I\left(\boldsymbol{\tau}_K - \boldsymbol{r}\right), \tag{3}$$

meaning that $\boldsymbol{r}$ is a linear, exponentially stable dynamic system driven by the foreign torque $\boldsymbol{\tau}_K$. Therefore, the residual vector remains at zero as long as the robot motors are generating the amount of torque that is expected for the given task and there is no external attack torque, while it increases or decreases in the presence of anomalies, converging to $\boldsymbol{\tau}_K$ at steady-state, with a time-constant that shortens as $\boldsymbol{K}_I$ increases. We remark that, for our purposes, the sign of $\boldsymbol{r}$ is irrelevant. Therefore, we define the *residual signal* as its euclidean norm $r(t) = \|\boldsymbol{r}(t)\|$.

Threshold design and filtering. In real-world scenarios, friction and other disturbances that were not explicitly considered in (1) also affect the robot. In our model, $\boldsymbol{\tau}_K$ absorbs their effect together with any actual foreign torque; thus, $\boldsymbol{\tau}_K$ does not remain at zero even in nominal conditions, and neither does r. To prevent false positives, attack detection requires a second step in which the residual signal is compared against a threshold $r_{max}(t)$, and an alarm is only

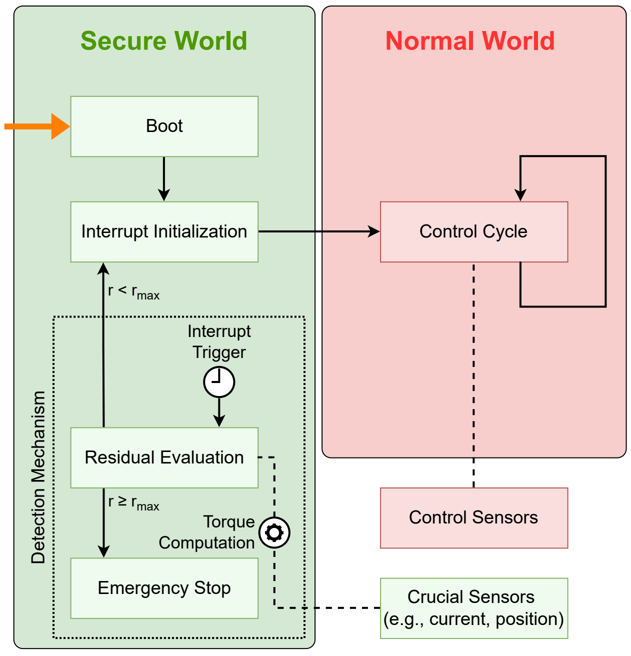


Figure 2: State diagram of the LLC with TrustZone-based attack detection.

raised if $r(t) \geq r_{max}(t)$. Defining a threshold involves making a trade-off between false positives and false negatives: if the threshold is set too low, small variations in the physical characteristics of the robot (e.g., due to wear) could cause a slight increase in the residual signal and trigger false alarms; on the other hand, if the threshold is set too high, stealthy attacks causing microscopic trajectory variations could go unnoticed or take too much time to be detected.

Threshold design can be made more efficient by filtering the residual signal before evaluating it. In fact, most environmental disturbances result in high-frequency oscillations, while signals due to motor torques are physically limited to lower frequencies. Thus, we consider and evaluate two filtering options: a low-pass filter can be used to smooth out the residual signal and set a more stringent threshold, or the 2-norm of the residual signal calculated on the full movement of the robot, defined as

$$r_{2n}(t) = \sqrt{\int_0^t r^2(q)\, dq}, \tag{4}$$

can also be used as the evaluation signal to be compared with the threshold, raising an alarm when $r_{2n}(t) \geq r_{max}(t)$. In this case, the integration operation is responsible for filtering out high-frequency oscillations. Regardless of the approach, the cutoff frequency must be evaluated carefully: if set too low, the filter's slow response could cause significant delays in the detection of attacks, while higher frequencies make the detection expose to noise and/or instantaneous sensors' jitters.

5.2 TrustZone-Enforced Security

We take advantage of TrustZone to guarantee the security of our attack detection system (ADS) so that its detection capabilities remain intact even in case the LLC software is completely compromised. We consider an LLC based on a TrustZone-enabled Arm Cortex-M microcontroller. No additional hardware, external components, or even modifications to the control code are required.

First, as shown in Figure 2, we ensure that all robot control code runs in the normal world, where it can be easily updated when necessary. All hardware interfaces and control sensors that the control cycle needs to interact with are also assigned to the normal world. Conversely, the attack detection code runs in the secure world, where it cannot be affected by a malicious actor that obtains control of the normal world and controller code. Similarly, sensor data crucial for the detection mechanism is configured to be accessible solely from the secure world, leveraging the TrustZone features outlined in Section 2.2, thereby maintaining its integrity. Indeed, the torque data necessary for the attack detection routine can be readily calculated using position and electrical current sensors, typically present in industrial robots. With these sensors directly interfacing with the microcontroller, bypassing interactions with other potentially vulnerable devices, their data can be exclusively routed to the secure world. This configuration makes it impractical for an attacker to intercept the sensor data before it is utilized in the detection routine.

If the control code requires this data, it is transferred via a secure interface to the normal world.

To call the attack detection algorithm, we set a secure timer interrupt to periodically trigger a context switch from the normal world to the secure world, where our attack detection routine runs. The timer interrupt duration is typically a multiple of the duration of a control cycle, depending on the desired detection frequency. Finally, we de-prioritize interrupts from the normal world so that non-trusted code has no way of hindering the attack detection process. The execution flow after our ADS has been implemented, see its pictorial representation in Figure 2, is as follows:

1. The controller boots into the secure world and configures its hardware, allocating resources to each world.
2. The controller schedules a timer interrupt for the next execution of the attack detection routine. Then, it performs a context switch to the normal world.
3. Non-secure control code executes and controls robot movements.
4. When the secure timer interrupt fires, the controller suspends the execution of the control code; it automatically switches to the secure world and starts executing the attack detection routine. It calculates the residual signal and evaluates it by comparing it with the threshold value. Data used here to execute detection is provided directly to the secure world.

5a. If no attacks are detected, go to step 2.
5b. If an attack is detected, the controller stops the robot motors and halts. To resume operation, it requires a restart.

We remark that, during nominal operation (i.e., steps 2–5a), the attack detection flow is completely deterministic. This renders it compatible with the real-time properties of the LLC, which require each task to have a predictable and bounded execution time. Finally, we observe that our approach only requires one context switch from the normal world to the secure world (and back) per attack detection cycle, minimizing its overhead.

6 Approach Evaluation

We implement our ADS using Matlab/Simulink, and we integrate it into the simulation of a 2-DOF robot manipulator operating in the xy space. The robot characteristics are listed in Table 1. In our evaluation, first, we test the effectiveness of our detection algorithm in recognizing attacks. Then, we evaluate the computational overhead introduced by our monitoring system. To do so, we export the Matlab/Simulink code in C and deploy it on an STM32 Nucleo board. Finally, we estimate the residual from typical operations of an actual robot, to assess the influence of the model uncertainties and how they would affect real-world functioning.

6.1 Evaluation Parameters

Table 1: Characteristics of the simulated robot.

Simulated robot		
Links	Length	1 m
	Mass	50 kg
	Center of Gravity	Midpoint
	Inertia Matrix	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 10 \end{bmatrix}$ kg · m^2
Motors	Gear Ratio	100
	Motor Inertia	0.005 kg · m^2 (motor 1) 0.002 kg · m^2 (motor 2)
P gain	Position Loop	$K_{pp1} = K_{pp2} = 20$
	Velocity Loop	$K_{pv1} = K_{pv2} = 2.2$
I gain	Velocity Loop	$K_{iv1} = 44$, $K_{iv2} = 17$
Friction model	$\tau_F(\dot{q}) = 10 + 45 \tanh\left(2.3\frac{\dot{q}}{38}\right) + 0.48\dot{q}$	

For the LLC we used a design that involves two independent axis controllers for the two robot links. Each one consists of a P controller, regulating joint position, and a PI controller, regulating joint velocity.

Table 2: Characteristics of the test trajectories.

Trajectory	Total Length	Segments length
Linear	1.70 m	1.70 m
Triangle	2.10 m	3x 0.70 m
Square	4.52 m	4x 1.13 m
S-shape	2.69 m	3x0.71 m and 2x0.28m, alternating

We design and run a set of experiments to evaluate, both in qualitative and quantitative terms, to what extent our system is effective in detecting attacks against the LLC. Our test cases consist of four trajectories inspired by real-world use cases of industrial robots, namely (1) straight line, inspired by pick-and-place tasks; (2) triangle and (3) square shapes, inspired by cutting tasks; (4) S-shape, inspired by painting tasks. Table 2 illustrates the characteristics of each trajectory. Each complete trajectory execution takes 10 seconds, and the velocity profile for each segment of all tests is trapezoidal.

Simulations are executed with a sampling time of 10 ms. To assess the effectiveness of our ADS, we simulate an attacker tampering with the LLC. We design several attacks for each of the three main categories described in Section 4.3, i.e.:

Table 3: Residual threshold equations.

Evaluation signal	Threshold equation
Plain residual	$r_{max}(t) = 1.0800 \times 10^{-4}t + 2.2000 \times 10^{-5}$
Filtered residual	$r_{max}(t) = 8.7620 \times 10^{-5}t + 1.0000 \times 10^{-5}$
Residual 2-norm	$r_{max}(t) = 2.5463 \times 10^{-5}t + 1.7500 \times 10^{-6}$

1) *Alteration of control loop parameters*: We increase the alteration progressively to evaluate if it affects detection, from slow and shallow oscillations to movements that exceed the physical capabilities of the robot.
   a) K_{pvx} divided by 10 ($K_{pv1} = 0.22$, $K_{pv2} = 0.22$) and K_{ivx} multiplied by 10 ($K_{iv1} = 440$, $K_{iv2} = 170$).
   b) K_{pvx} divided by 5 ($K_{pv1} = 0.44$, $K_{pv2} = 0.44$) and K_{ivx} multiplied by 5 ($K_{iv1} = 220$, $K_{iv2} = 85$).
   c) K_{pvx} multiplied by 10 ($K_{pv1} = 22$, $K_{pv2} = 22$) and K_{ivx} divided by 10 ($K_{iv1} = 4.4$, $K_{iv2} = 1.7$).
   d) K_{pvx} multiplied by 40 ($K_{pv1} = 88$, $K_{pv2} = 88$) and K_{ivx} divided by 40 ($K_{iv1} = 1.1$, $K_{iv2} = 0.425$).
2) *Alteration of input processing*:
   a) *Injection of noise to the position signal*: We add increasing noise to the signal input to simulate attacks aimed at generating minor modifications.
      i) *Gaussian distribution with* $\mu = 0, \sigma = 0.05$.
      ii) *Gaussian distribution with* $\mu = 0, \sigma = 0.5$.
   b) *Skewing position signal*: We modify the input position signal either of a fixed value or with a gradual increase, to evaluate detection of abrupt or gradual modifications to the input of the robot closed loop.
      i) *Fixed skew by* $[0.25, 0.25]$ *rad*.
      ii) *Fixed skew by* $[0.50, 0.50]$ *rad*.
      iii) *Increasing skew with equation* $y = 5t$.
      iv) *Increasing skew with equation* $y = 10t$.
3) *Alteration of output processing*:
   a) *Injection of noise to the position signal*: We add different noise values to the output position signal.
      i) *Uniform distribution in [-100,100] rad*.
      ii) *Uniform distribution in [-200,200] rad*.
   b) *Torque Output changes*: The torque sent to the motors stops being the one generated by the controller logic and is maintained at a constant, realistic value.
      i) *Torque maintained at [30,30] Nm*.
      ii) *Torque maintained at [60,60] Nm*.

We also configure our ADS to use, one at a time, the three evaluation signals described in Section 5.1 so that their performances can be compared and evaluated separately: "plain" residual, low-pass-filtered residual (filter cutoff frequency: 10 Hz), residual 2-norm. For each evaluation signal, we empirically define a threshold function such that non-zero values caused by friction and other modeling imperfections will not trigger an alarm for any test case. Table 3 presents the equations of the functions we defined.

We evaluate the results using two metrics. Time to detection (TTD) measures the time elapsed between the start of the attack and its detection. Maximum trajectory deviation (MTD) represents the maximum distance the manipulator reaches from the nominal trajectory before the attack is detected.

Table 4: Attack detection experimental results. The letter and number that represent attacks refer to section 6.1. For each test instance, the table displays the TTD and MTD. A missing TTD indicates that the detection system never raised an alert. P.R. stands for Plain Resolution, L-P for Low-Pass, and 2-N for 2-Norm.

Attack	Metric	Linear Movement			Triangular Movement			Square Movement			S-shape Movement		
		P.R.	L-P	2-N	P.R.	L-P	2-N	P.R.	L-P	2-N	P.R.	L-P	2-N
None		—	—	—	—	—	—	—	—	—	—	—	—
1a: $K_{pv}/10$	s	—	—	9.38	3.13	0.64	0.62	0.07	0.11	0.09	0.07	0.11	0.09
$K_{iv}x10$	mm	22.82	22.82	17.57	16.60	6.73	6.73	0.57	1.98	1.32	0.56	1.96	1.30
1b: $K_{pv}/5$	s	—	—	—	—	—	—	0.07	0.13	0.11s	0.07	0.13	0.11
$K_{iv}x5$	mm	1.25	1.25	1.25	5.30	5.30	5.30	0.49	2.36	1.68	0.48	2.33	1.66
1c: $K_{pv}x10$	s	—	—	—	0.33	0.27	0.27	0.07	0.13	0.11	0.07	0.13	0.11
$K_{iv}/10$	mm	10.59	10.59	10.59	9.78	6.56	6.56	0.61	3.12	2.18	0.60	3.08	2.15
1d: $K_{pv}x40$	s	7.24	7.09	3.00	0.22	0.23	0.23	0.06	0.12	0.11	0.06	0.12	0.11
$K_{iv}/40$	mm	88.47	88.47	43.99	4.68	5.48	5.48	0.56	2.84	2.25	0.55	2.80	2.22
2(a)i: Gaussian noise	s	0.01	0.08	0.01	0.01	0.07	0.01	0.02	0.06	0.04	0.01	0.08	0.04
$\mu = 0, \sigma = 0.05$	mm	0.23	1.77	0.46	0.20	0.47	1.07	0.07	1.01	1.01	0.11	0.64	1.07
2(a)ii: Gaussian noise	s	0.02	0.05	0.01	0.01	0.03	0.01	0.01	0.05	0.01	0.02	0.06	0.03
$\mu = 0, \sigma = 0.5$	mm	1.59	1.05	0.89	0.04	2.02	1.11	0.06	1.77	0.41	0.07	0.74	2.07
2(b)i: Fixed skew	s	0.01	0.07	0.03	0.01	0.07	0.03	0.01	0.07	0.03	0.01	0.07	0.03
$[0.25, 0.25]$	mm	0.21	2.26	1.08	0.21	2.26	1.08	0.21	2.26	1.08	0.21	2.26	1.08
2(b)ii: Fixed skew	s	0.01	0.03	0.02	0.01	0.03	0.02	0.01	0.03	0.02	0.01	0.03	0.02
$[0.50, 0.50]$	mm	0.42	2.16	1.26	0.42	2.16	1.26	0.42	2.16	1.26	0.42	2.16	1.26
2(b)iii: Increasing skew	s	—	—	—	—	—	—	2.57	2.62	2.35	—	—	2.09
$y = 5t$	mm	139.90	139.90	139.90	53.51	53.51	53.51	26.63	27.18	48.65	89.48	89.48	33.07
2(b)iv: Increasing skew	s	—	—	—	—	—	0.08	0.06	2.58	0.07	0.06	—	0.07
$y = 10t$	mm	280.80	280.80	280.80	108.20	108.20	1.74	0.26	53.52	1.31	0.26	181.20	1.31
3(a)i: Uniform noise	s	0.56	0.19	0.83	0.24	0.42	0.40	0.06	0.19	0.13	0.05	0.47	0.17
[-100,100] rad	mm	3.95	3.50	4.78	1.71	3.03	3.35	0.53	2.54	2.37	0.44	4.36	1.13
3(a)ii: Uniform noise	s	0.04	0.08	0.31	0.03	0.10	0.15	0.03	0.08	0.08	0.01	0.12	0.11
[-200,200] rad	mm	1.47	3.37	3.73	0.25	3.87	1.27	0.03	3.32	1.41	0.19	2.28	1.40
3(b)i: Torque Output	s	0.23	0.30	0.39	0.27	0.32	0.39	0.15	0.19	0.17	0.15	0.22	0.24
[30-30] Nm	mm	17.71	31.37	51.99	22.29	32.54	49.49	9.50	16.44	12.17	6.52	13.81	16.54
3(b)ii: Torque Output	s	0.12	0.19	0.23	0.13	0.19	0.23	0.09	0.15	0.13	0.16	0.21	0.24
[60-60] Nm	mm	10.00	24.51	35.86	11.06	24.06	34.27	5.79	15.38	11.43	14.58	28.08	33.79

6.2 Attack Detection Evaluation

The numerical results of each test run are provided in Table 4. The first line shows how the detection system does not trigger false positives if no attack is implemented. Results show that the system quickly detects all attacks that provoke high-frequency oscillations or abrupt movements of the manipulator, even if they have a minimal influence on the robot's trajectory. For instance, depending on the trajectory and evaluation signal, the MTD for attacks 2(a)i 2(a)ii 2(b)i 2(b)ii (Fig 3b) range from 0.04–2.26 mm. In fact, When adding noise to the position signal, the LLC constantly attempts to compensate with short bursts of motor torque, resulting in high frequency and low-amplitude oscillations of the robot arm. Although the robot deviates of a tiny amount from the nominal trajectory, high frequency oscillations cause a large residual increase which is always easily detected.

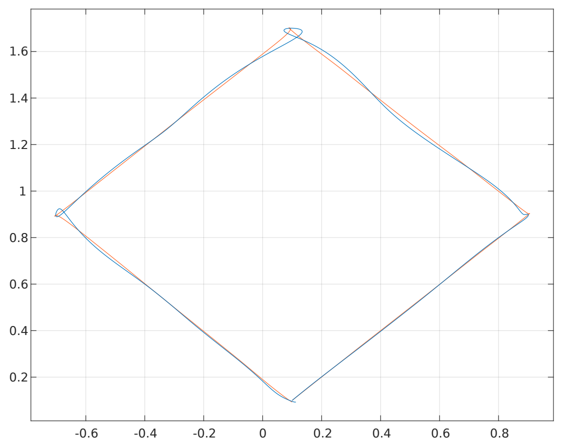
All attacks from category 1 cause relatively low-frequency oscillations, whose magnitude increases over time; as a direct consequence, we observe larger TTDs. Attacks 1a, 1b, and 1c (Fig 3a) are undetected during the straight line trajectory aside for one case, due to the slower movements of the joints. Noticeably, square and s-shaped movements lead to effective detection, thanks to the increased speed of the motors, with a detection time from 0.07 s up to 0.13 s depending on the filter.

Attacks 2(b)iii and 2(b)iv (Fig 3c) highlight the main limitation of our approach, already hinted by other undetected attacks: this type of attack causes the position signal to gradually and continuously drift away from the nominal value, resulting in a constant offset of the residual signal. The residual offset is directly proportional to the error increasing rate, making the attack difficult to detect. The attack is mostly undetected aside for the square shaped trajectory, with an MTD that varies from 0.26 mm to 53.52 mm when detected, and which reaches up to 280.80 mm while being undetected, due to the extremely slow deviation from the expected path.

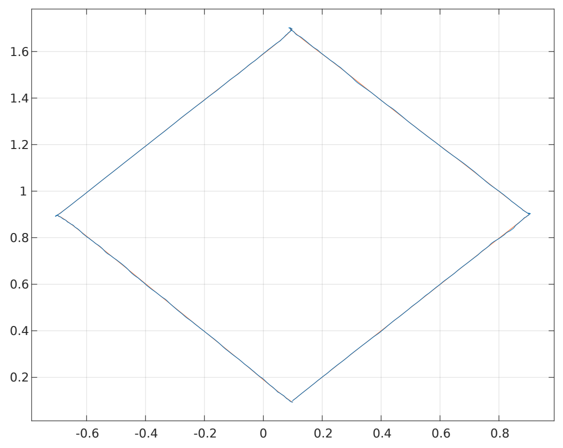
Finally, attacks 3a, 3b (Fig 3d), predictably, are easily detected. The attack, in fact, heavily modifies the residual as soon as it starts, rapidly triggering detection. This is true for both the standard residual and the filtered ones.

As expected, the detection time is overall higher when using the filtered residual and the 2-norm of the residual as evaluation signals. However, the results also show that there are instances where the plain residual does not detect attacks which are instead detected by the 2-norm filter (1a,2(b)iii,2(b)iv). Interestingly, the low-pass filter instead does not present any advantage from the plain resolution signal.

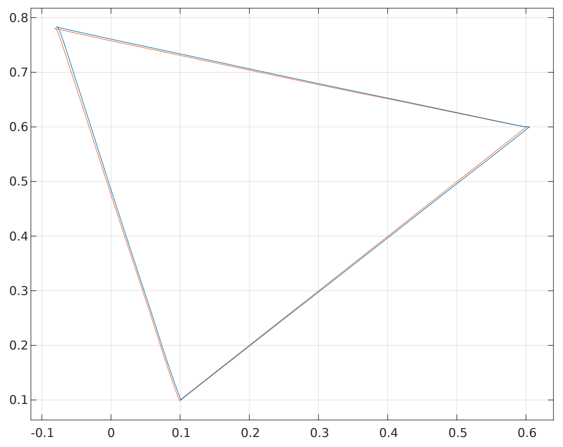
Overall, experimental results show that our ADS is very effective at detecting attacks that affect the manipulator with high-frequency oscillations, abrupt movements, or direction changes, even when their influence on the robot's trajectory is in the order of a few millimeters or less. With this kind of attack, our system can protect industrial robots from both critical threat scenarios described in Section 4.1: physical damage and production sabotage. Slow-acting attacks, whose effects span over periods of tens of seconds, are more difficult to detect and result in some false negatives.



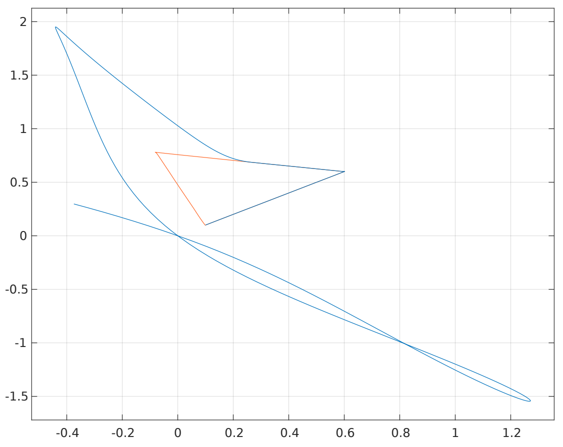
(a) Alteration of control loop parameters - K_{pvx} divided by 10 ($K_{pv1} = 0.22$, $K_{pv2} = 0.22$) and K_{ivx} multiplied by 10 ($K_{iv1} = 440$, $K_{iv2} = 170$) - square movement.



(b) Alteration of input processing - Gaussian distribution with $\mu = 0, \sigma = 0.05$ - square movement.



(c) Alteration of input processing - Increasing skew with equation y=5t - triangular movement.



(d) Alteration of output processing - Torque output mantained at [30-30] Nm - triangular movement.

Figure 3: Effects of attacks on the slip of the wheels when bounds are crossed.

Table 5: TrustZone execution time requirements.

Operation	Time requirement
Context switch to the secure world	0.004 ms
Attack detection routine	1.396 ms
Context switch to the normal world	0.008 ms
Total	1.408 ms

We argue, though, that such attacks pose a limited threat with respect to physical damage, although they could still be successful at injecting faults into the product.

6.3 TrustZone Timing Evaluation

The remainder of our evaluation aims to measure the time efficiency of the proposed solution when deployed on a TrustZone-enabled physical development board in order to estimate its impact on the real-time properties of the LLC.

We use the MATLAB and Simulink code generation functionalities to export the C code of our simulation. Using *STM32CubeIDE*, we bring the changes needed in order to specify which procedures have to be executed in the secure world, and we add time measuring function calls where necessary. We upload and execute the code on an STM32 Nucleo L552ZE-Q development board, recording timing results.

Table 5 illustrates the overhead introduced by our solution in terms of time requirements. It can be noticed that the total latency caused by world switches has a very limited impact, and our system spends almost the totality of the time on the actual execution of the attack detection routine. Execution times obviously depend on the hardware characteristics of each microcontroller; our numeric results can be scaled according to the processor speed to obtain an estimate of time requirements on other devices. Implementing our solution on a specific LLC without affecting its real-time properties requires it to be able to carry out both the usual control computations and the attack detection routine within the duration of a single control cycle. Based on hard-

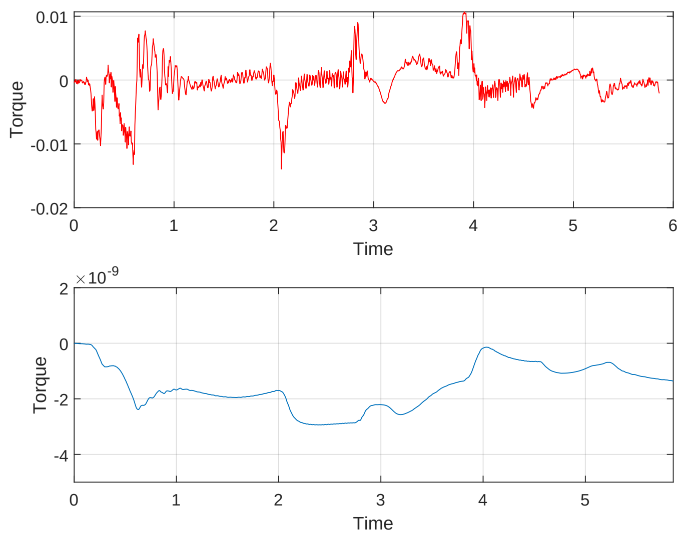


Figure 4: In red the difference between the real-world and estimated torque for T1, in blue the corresponding residual.

ware characteristics, attack detection can be either set up to run at each execution cycle or to only run every n cycles. The former option guarantees the best performances in terms of TTD and MTD, while the latter trades off detection speed for a lighter impact on the controller's resources.

6.4 Empirical Nominal Residual Estimation

To assess the influence of model uncertainties and how they would affect the real-world functioning of the approach on the industrial robot used in our experiment, we calculate the residual from the typical operations of an actual robot.

To achieve this, we measured the torque of the six motors of the robotic arm controller and then compared these measurements with their simulated counterparts. Following this, we calculated the residual of the difference between the registered torque and the simulated one. Figure 4 displays the results of the comparison. In the figure, the discrepancy between the actual and estimated torque is indicated in blue, and the corresponding residual for motor T1 is highlighted in red. Data for other motors, which follow similar patterns, are not included for the sake of clarity. Although a discrepancy exists between the simulated and actual values during regular operations, as depicted by the blue line, the calculated residual from these values is minimal. This is further shown in Table 6, where we provide the mean, variance, and maximum value of the residual for each motor. Upon comparing with the detection thresholds estimated in this work (see Table 3), it's clear that the residual produced by the real robot is significantly lower — three orders of magnitude less — than our estimated thresholds. Consequently, these noises are unlikely to interfere with the identification of anomalies.

Table 6: Mean, Variance, and Maximum value of the residual calculated on the real-world motor torques T1-T6 of the six motors of the robotic arm.

	Scaling	T1	T2	T3	T4	T5	T6
Mean	10^{-8}	0.1545	0.2416	0.3397	0.0363	0.1180	0.0317
Variance	10^{-17}	0.0633	0.1598	0.2860	0.0063	0.0817	0.0099
Max	10^{-8}	0.2939	0.4242	0.6135	0.0932	0.2292	0.0831

7 Conclusions

In this paper, we presented Janus, a novel LLC attack detection approach based on the *state observers* strategy and implemented in the robot controller system. Differently from existing approaches, Janus exploits the Arm trusted execution environment (TEE) to guarantee the integrity of the detection capabilities even in case the LLC software is completely compromised, while being installed on the same controller and not requiring any additional hardware. We evaluated the system performance by implementing it the Matlab/Simulink environment and on a physical, TrustZone-equipped development board. Experimental results showed that our solution is very effective at detecting fast-acting attacks, while it has limitations in detecting slowly-acting ones. In addition, we demonstrated that the overhead caused by TrustZone technology has a negligible impact on the final performance.

Given the widespread implementation of TEE technologies in modern microprocessors, we envision the application of the same TEE-based monitoring mechanism to other cyberphysical domains, such as Industry 4.0 machinery, medical devices, and intelligent transportation systems. However, given the peculiarities of the different domains, future works should focus on studying the feasibility of embedding existing detection systems into TEE-based technologies. Moreover, besides detecting anomalous behavior, future works should focus on implementing various context-dependent reaction strategies, e.g., resetting the microprocessor or switching to a different controller. This is especially relevant in domains in which attacks may impact the safety of living beings cooperating with industrial systems.

In order to overcome the limitation related to the lower performance obtained with slow-acting attacks, future work will focus on improving the threshold generation process. The current solution used a straightforward, empirical method for its definition. However, it could be worth developing an *adaptive* thresholding technique that dynamically adjusts the detection sensitivity based on the characteristics of each phase of the robot motion, inspired by the research works in the field of fault detection and identification, see e.g., [34, 35, 36].

Finally, future works may expand the threat model considering adversarial attacks (i.e., adversaries that try to evade, poison, or extract knowledge from detection models under attacks). For instance, an interesting use case scenario may be represented by attackers that exploit the determinism of the system under analysis.

Acknowledgment

This work is supported by the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from Next-Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004). CUP MICS D43C22003120001.

REFERENCES

[1] International Federation of Robotics, "World robotics," https://ifr.org/worldrobotics/, 2021. 1

[2] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, pp. 1–19, 2017. 1

[3] A. C. Simões, A. Pinto, J. Santos, S. Pinheiro, and D. Romero, "Designing human-robot collaboration (hrc) workspaces in industrial settings: A systematic literature review," *Journal of Manufacturing Systems*, vol. 62, pp. 28–43, 2022. 1

[4] F. Vicentini, "Terminology in safety of collaborative robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 63, p. 101921, 2020. 1

[5] N. Lucci, B. Lacevic, A. M. Zanchettin, and P. Rocco, "Combining speed and separation monitoring with power and force limiting for safe collaborative robotics applications," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6121–6128, 2020. 1

[6] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero, "An experimental security analysis of an industrial robot controller," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 268–286. 1, 3

[7] M. Pogliani, D. Quarta, M. Polino, M. Vittone, F. Maggi, and S. Zanero, "Security of controlled manufacturing systems in the connected factory: the case of industrial robots," *Journal of Computer Virology and Hacking Techniques*, vol. 15, 09 2019. 1, 3

[8] F. Maggi, D. Quarta, M. Pogliani, M. Polino, A. M. Zanchettin, and S. Zanero, "Rogue robots: Testing the limits of an industrial robot's security," https://documents.trendmicro.com/assets/wp/wp-industrial-robot-security.pdf. 1

[9] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017. 1, 4

[10] Texas Instruments, "An engineer's guide to industrial robot designs," https://www.ti.com/lit/eb/ssiy006/ssiy006.pdf, 2020. 2

[11] M. W. Spong, S. Hutchinson, M. Vidyasagar *et al.*, *Robot modeling and control*. wiley New York, 2006, vol. 3. 2, 4

[12] *OSHA Technical Manual (OTM)*, United States Department of Labor, Occupational Safety & Health Administration, 2021. [Online]. Available: https://www.osha.gov/otm/section-4-safety-hazards/chapter-4 2

[13] S. Pinto and N. Santos, "Demystifying Arm TrustZone: A comprehensive survey," *ACM Comput. Surv.*, vol. 51, no. 6, January 2019. [Online]. Available: https://doi.org/10.1145/3291047 2

[14] L. Luo, Y. Zhang, C. White, B. Keating, B. Pearson, X. Shao, Z. Ling, H. Yu, C. C. Zou, and X. Fu, "On security of trustzone-m-based iot systems," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9683–9699, 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2022.3144405 2

[15] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Comput. Surv.*, vol. 51, no. 4, July 2018. [Online]. Available: https://doi.org/10.1145/3203245 2

[16] A. de Faveri Tron, S. Longari, M. Carminati, M. Polino, and S. Zanero, "Canflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM, 2022, pp. 711–723. [Online]. Available: https://doi.org/10.1145/3548606.3560618 2

[17] A. Erba, R. Taormina, S. Galelli, M. Pogliani, M. Carminati, S. Zanero, and N. O. Tippenhauer, "Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems," in *ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020*. ACM, 2020, pp. 480–495. [Online]. Available: https://doi.org/10.1145/3427228.3427660 2

[18] M. S. Chong, M. Wakaiki, and J. P. Hespanha, "Observability of linear systems under adversarial attacks," in *2015 American Control Conference (ACC)*, 2015, pp. 2439–2444. 3

[19] Y. Shoukry, M. Chong, M. Wakaiki, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, J. P. Hespanha, and P. Tabuada, "SMT-based observer design for cyber-physical systems under sensor attacks," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, 2016, pp. 1–10. 3

[20] S. Mishra, Y. Shoukry, N. Karamchandani, S. N. Diggavi, and P. Tabuada, "Secure state estimation against sensor attacks in the presence of noise," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 49–59, 2017. 3

[21] C. Murguia and J. Ruths, "Characterization of a CUSUM model-based sensor attack detector," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 1303–1309. 3

[22] Y. Mo, S. Weerakkody, and B. Sinopoli, "Physical authentication of control systems: Designing watermarked control inputs to detect counterfeit sensor outputs," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 93–109, 2015. 3

[23] Y. Shoukry, P. Martin, Y. Yona, S. Diggavi, and M. Srivastava, "PyCRA: Physical challenge-response authentication for active sensors under spoofing attacks," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1004–1015. [Online]. Available: https://doi.org/10.1145/2810103.2813679 3

[24] D. Muniraj and M. Farhood, "Detection and mitigation of actuator attacks on small unmanned aircraft systems," *Control Engineering Practice*, vol. 83, pp. 188–202,

2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066118306804 3

[25] X. Jin, W. M. Haddad, and T. Yucelen, "An adaptive control architecture for mitigating sensor and actuator attacks in cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 11, pp. 6058–6064, 2017. 3

[26] V. Narayanan and R. B. Bobba, "Learning based anomaly detection for industrial arm applications," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS-SPC '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 13–23. [Online]. Available: https://doi.org/10.1145/3264888.3264894 3

[27] H. R. Ghaeini, M. Chan, R. Bahmani, F. Brasser and, L. Garcia, J. Zhou, A.-R. Sadeghi, N. O. Tippenhauer, and S. Zonouz, "PAtt: Physics-based attestation of control systems," in *Proceedings of International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, September 2019. 3

[28] A. Munawar, P. Vinayavekhin, and G. De Magistris, "Spatio-temporal anomaly detection for industrial robots through prediction in unsupervised feature space," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 1017–1025. 3

[29] H. Pu, L. He, C. Zhao, D. K. Y. Yau, P. Cheng, and J. Chen, *Detecting Replay Attacks against Industrial Robots via Power Fingerprinting*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 285–297. [Online]. Available: https://doi.org/10.1145/3384419.3430775 3

[30] F. Abdi, C.-Y. Chen, M. Hasan, S. Liu, S. Mohan, and M. Caccamo, "Guaranteed physical security with restart-based design for cyber-physical systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018, pp. 10–21. 3

[31] M. Hasan and S. Mohan, "Protecting actuators in safety-critical IoT systems from control spoofing attacks," *CoRR*, vol. abs/1908.09444, 2019. [Online]. Available: http://arxiv.org/abs/1908.09444 3

[32] D. Kleidermacher and M. Kleidermacher, *Embedded systems security: practical methods for safe and secure software and systems development*. Elsevier, 2012. 4

[33] A. De Luca and R. Mattone, "Actuator failure detection and isolation using generalized momenta," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1, 2003, pp. 634–639. 4

[34] Y. Baek and M. Jo, "Adaptive threshold generation for fault detection with high dependability for cyber-physical systems," *Applied Sciences*, vol. 8, no. 11, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/11/2235 9

[35] S.-A. Raka and C. Combastel, "Fault detection based on robust adaptive thresholds: A dynamic interval approach," *Annual Reviews in Control*, vol. 37, no. 1, pp. 119–128, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578813000102 9

[36] L. M. Ho, "Application of adaptive thresholds in robust fault detection of an electro-mechanical single-wheel steering actuator," *IFAC Proceedings Volumes*, vol. 45, pp. 259–264, 2012. 9

Stefano Longari Stefano Longari received a Ph.D. in Information Technology from Politecnico di Milano, where he currently works as a Postdoctoral Researcher in NECST laboratory as part of the System Security group inside the Dipartimento di Elettronica, Informazione e Bioingegneria. His main research focus is automotive security and the use of novel technologies such as machine learning intrusion detection to secure the development of smart mobility and smart city ecosystems. Since 2020 he is a member of the Automotive Security Research Group.

Jacopo Jannone received a M.Sc. degree in Computer Science and Engineering from Politecnico di Milano in 2022. He is currently working at Secure Network as an information security specialist, where he routinely evaluates the security of hardware devices and embedded systems. His activities include penetration tests of mobile, web, and native applications, source code reviews, and security assessments of corporate networks.

Mario Polino has got his PhD (2017) in Computer Security from Politecnico di Milano. Currently, Mario is an Assistant Professor at Politecnico di Milano. His PhD was mainly based on Malware Analysis and automation of behavior extraction. Besides malware analysis, Mario is interested in several aspects of Computer Security; his works range from the study of Cyber-Physical systems to Banking Fraud Detection going through Android Security. Even more, he is interested in Binary analysis and Exploitation techniques. Mario spends most of his free time playing Capture the Flag Competition with "Tower fo Hanoi" and "mhackeroni".

Michele Carminati Michele Carminati received his Ph.D. degree cum laude in Information Technology from Politecnico di Milano in Italy, where he is currently an Assistant Professor and a Postdoctoral Researcher working at NECST laboratory as part of the System Security group inside the Dipartimento di Elettronica, Informazione e Bioingegneria. His research revolves around the application of machine learning methods in various cybersecurity-related fields, ranging from cyber-physical and automotive systems to binary and malware analysis, going through anomaly and intrusion detection. He is actively involved in research projects funded by the European Union, and he is also co-founder of Banksealer, a Fintech spinoff of Politecnico di Milano.

Andrea Zanchettin Andrea M. Zanchettin was born in Cremona (Italy) in 1983. He received his PhD in Information Technology, with honour, from Politecnico di Milano in 2012. From 2014 until 2019 he has been an assistant professor at DEIB where he is now a tenured Associate Professor. His research interests are mechatronic systems, automatic control, and intelligent human-robot interaction. Since 2017, Andrea Zanchettin has been co-founder and co-chair of the IEEE RAS Technical Committee on Collaborative Automation for Flexible Manufacturing (CAFM). Dr. Zanchettin is also chair of the Italian Chapter of the IEEE RAS (I-RAS), as well as vice president for industrial activities of the Italian Institute of Robotics and Intelligent Machines (I-RIM). He is also co-founder and member of the Board of Directors of Smart Robots, a spin-off company of Politecnico di Milano.

Mara Tanelli Mara Tanelli (M'05, SM'12) is currently a Full Professor of Automatic Control at the Politecnico di Milano, where she earner her Ph.D. in Information Engineering with honors in 2007. Her main research interests are in Automotive Control, Smart Mobility and Industrial Analytics. She is co-author of more than 170 peer-reviewed publications in these research areas. Prof. Tanelli is a member of the Conference Editorial Board of the IEEE Control Systems Society (CSS). She is AE for the IEEE Transactions on Control Systems Technology and for the IEEE Transactions on Human-Machine Systems. Since 2021 she is Chair of the Technical Committee (TC) on Automotive Controls of the IEEE CSS and vice-chair for publications of the corresponding IFAC TC.

Stefano Zanero Stefano Zanero (M'03, SM'10) received a Ph.D. degree in Computer Engineering from Politecnico di Milano, where he is currently a Full Professor with the Dipartimento di Elettronica, Informazione e Bioingegneria. His research focuses on cybersecurity, in particular cyber-physical systems, malware analysis, and fraud and anomaly detection through the use of machine learning. Prof. Zanero is a Distinguished Contributor of the IEEE Computer Society, where he currently serves on the Board of Governors.