PAtt: Physics-based Attestation of Control Systems

Hamid Reza Ghaeini[1], Matthew Chan[2], Raad Bahmani[3], Ferdinand Brasser[3], Luis Garcia[4], Jianying Zhou[1], Ahmad-Reza Sadeghi[3], Nils Ole Tippenhauer[5], and Saman Zonouz[2]

[1]Singapore University of Technology and Design, ghaeini@acm.org, jianying_zhou@sutd.edu.sg
[2]Rutgers University, {matthew.chan, saman.zonouz}@rutgers.edu
[3]TU Darmstadt, {raad.bahmani, ferdinand.brasser, ahmad.sadeghi}@trust.tu-darmstadt.de
[4]University of California, Los Angeles, garcialuis@ucla.edu
[5]CISPA Helmholtz Center for Information Security, tippenhauer@cispa.saarland

Abstract

Ensuring the integrity of embedded programmable logic controllers (PLCs) is critical for the safe operation of industrial control systems. In particular, a cyber-attack could manipulate control logic running on the PLCs to bring the process of safety-critical application into unsafe states. Unfortunately, PLCs are typically not equipped with hardware support that allows the use of techniques such as remote attestation to verify the integrity of the logic code. In addition, so far remote attestation is not able to verify the integrity of the physical process controlled by the PLC.

In this work, we present PAtt, a system that combines remote software attestation with control process validation. PAtt leverages operation permutations—subtle changes in the operation sequences based on integrity measurements—which do not affect the physical process but yield unique traces of sensor readings during execution. By encoding integrity measurements of the PLC's memory state (software and data) into its control operation, our system allows us to remotely verify the integrity of the control logic based on the resulting sensor traces. We implement the proposed system on a real PLC, controlling a robot arm, and demonstrate its feasibility. Our implementation enables the detection of attackers that manipulate the PLC logic to change process state and/or report spoofed sensor readings (with an accuracy of 97% against tested attacks).

1 Introduction

Industrial control systems (ICS) are a class of cyber-physical systems (CPS) that typically consist of industrial controllers sensing and actuating safety-critical applications, e.g., the power grid, water treatment plants, and factory automation [58]. In particular, ICS typically consist of programmable logic controllers (PLCs), which are embedded systems that act as a reliable and re-programmable cyber-physical interface between a monitoring entity, i.e., the Supervisory Control and Data Acquisition (SCADA) center, and field-level devices, i.e., sensors and actuators that interface directly with the physical environment. As such, the security of these controllers is critical for ensuring the safe operation of the ICS [40].

Because of the safety-critical nature of PLCs, they have been typically targeted by nation-state malware, such as the infamous Stuxnet worm [18] against uranium enrichment facilities in Iran and the BlackEnergy crimeware [17] that targeted Ukrainian electric power and train railway systems [14, 47]. These attacks typically target the application-layer software, so-called *control logic*, due to the lack of security features in legacy industrial protocols. Although it has been shown that attackers can implement firmware-level attacks [6, 20], these attacks have been shown to be much more challenging to implement as they require a much more concerted effort for stealthiness.

Although these attacks are understood, they are challenging to defend against as security solutions need to be employed for legacy systems with fixed hardware. Currently, PLCs do not have hardware support to provide a hardware root-of-trust. Physical Unclonable Functions (PUFs) have been proposed in the past to enable software attestation for resource-constrained devices, but such modules are also not yet available for industrial devices. Existing integrity checks in industrial devices are limited to checksums that are preloaded onto the device when the program is initially loaded. In prior works, several solutions have been presented to either test the code that is being loaded onto the device [41] or verify the cyber and physical behavior of the overall CPS [2, 21, 22, 60]. However, these solutions treat the PLCs as black boxes and are not able to monitor the internal states at run-time. We conclude that a comprehensive solution to enable remote attestation of the logic running on a PLC is missing.

In this paper, we present PAtt, a remote attestation technique that combines software remote attestation with a physical PUF to attest the control-logic code is running on PLCs without trusted hardware. PAtt allows a verifier to challenge a PLC to generate an attestation report in the form of sensor values which are affected by a series of actuation commands (based on the challenge and checksums over the PLC logic).

The verifier can then attest the logic code integrity based on the measurement of the PLC memory, as well as the authenticity of the reply through the sensor values (similar to a PUF). In particular, we also show that PAtt can detect attempts of the attacker to replay the sensor readings with an accuracy of 97%. PAtt detects those manipulations based on an anomaly detector that is trained with data resulting from normal operation and does not need to be trained with prior attack examples.

Contributions. We summarize our contributions as follows.

- We present PAtt, a novel remote attestation technique for PLCs that combines software remote attestation with a PUF-like use of the physical process to attest the software and process state of the PLC.
- We theoretically investigate the performance of the proposed system and show that it is resilient against replay attacks which provide the sensor reading from a sensor record table. PAtt does not need to have prior knowledge of possible attacks, and it only requires the normal operational data during the training phase of the detector.
- We then implement and practically evaluate PAtt on a real ICS—a robotic arm in the context of a safety-critical process—and show that PAtt can detect the tested attacks with an accuracy of 97%.

The rest of the paper is structured as follows. First, we provide a background on previous techniques in remote attestation for CPS in Section 2. We describe the system model and design of PAtt in Section 3. Details on our implementation are provided in Section 4. We present our evaluation results in Section 5. We discuss the applicability of PAtt in Section 6 and we summarize related work in Section 7. Finally, we conclude in Section 8.

2 Background

In this section, we first provide an introduction to programmable logic controllers (PLCs) in the context of cyber-physical industrial control systems (ICS) as well as their security limitations. We then provide background on previous works in attestation of cyber-physical systems.

Industrial Control Systems. Modern industrial control systems consist of three major levels [29]:

- Supervisory Control And Data Acquisition (SCADA): this level of the ICS is mainly used for the control and monitoring of industrial process that may consist of large-scale geographical distributed computers. Five major components of the SCADA are the human-machine interface (HMI), data acquisition server, historian, engineer workstations, and remote workstation.
- Programmable Logic Controllers (PLC): The local control component that is mostly designed for managing a single process in ICS. PLCs are industrial computers that are developed for handling the process level devices like sensors and actuators.
- Fieldbus: The physical elements like sensors and actuators are connected to the PLC at this level. Most of the recent Fieldbus implementations use the Device Level Ring (DLR) with two redundant PLCs and a ring topology between those PLCs and physical elements.

In cyber-physical systems, the term Programmable Logic Controller refers to computing devices, which control the industrial appliances. Each PLC consists of (1) computing modules, which are designed to perform industrial processes reliably, (2) input modules translating analog physical inputs to digital values, and (3) outputs modules, which map the PLC's digital outputs to analog physical outputs.

In a PLC, the next system state is computed based on the current state measured by the input and output modules. The main part of the logic executed on a PLC (*control logic*) is programmed in special-purpose industrial languages, e.g., ladder logic, designed to guarantee a reliable transfer between system states [9]. Control logic is compiled at a SCADA server and downloaded to the PLC. The PLC runs this program to perform a control task by processing a set of inputs, received from physical sensors, and generating outputs to be interpreted by actuators. Control logic runs on top of a privileged software layer, e.g., a real-time operating system (RTOS), which provides the required services. The control logic consists of *function blocks*, *data blocks*, and *organization blocks*. Function blocks contain reusable functions and data blocks include data structures holding *global* or *local* variables used in the control logic. Organization blocks serve as the entry point of a PLC program and execute in a fixed time interval, known as *scan cycles*.

Remote Attestation. Remote attestation is a technique, which provides an external verifier with proofs on the integrity of a system's software state. It is termed *software-based attestation* when the proof of integrity is generated with no hardware aid. This form of attestation is based on strong assumptions regarding the time and the authenticity of the communication channel between the system and the external verifier [3, 49]. The limitations of software-based attestation can be overcome by using trusted hardware. A software, protected by trusted hardware, could measure the systems software stack and authenticate the measurement using a secret key which is likewise protected by the trusted hardware. Attestation techniques can be used together with other security solutions like Transport Layer Security (TLS) to prevent eavesdropping and Man-in-the-Middle scenarios.

Physically Unclonable Functions. Physically Unclonable Functions (PUFs) are like a physical fingerprint used in semiconductors to generate key information with a level of randomness from a complex physical system [27]. PUFs leverage

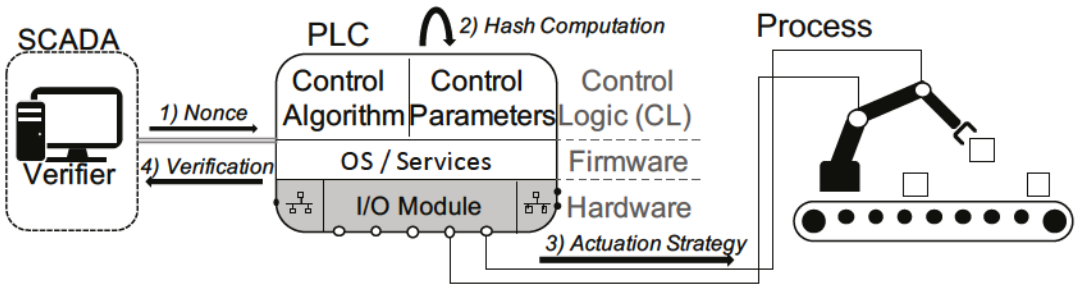


Figure 1: Overview of steps in PAtt framework.

unpredictable physical variations that occur naturally during semiconductor manufacturing. PUFs are used together with hash functions in many cryptographic applications [32, 50]. The latest versions of PUFs are equipped in integrated circuits and used in different security applications like software licensing. Recent industrial control systems do not include PUF-based integrated circuits inside the PLCs. However, there are some proposals to use the physical process as a PUF [62].

3 PAtt: Physics-based Attestation

PAtt is designed to allow remote attestation of logic code running on a PLC without a traditional trust anchor (such as a TPM or PUF). The devices that we target—PLCs in existing and legacy systems—are not usually equipped with trusted computing hardware to enable a hardware-based remote attestation process. While recent versions of PLC firmware (e.g., in the Siemens S7 series) include APIs that can be used for checksum generation over data blocks of control logic, the challenge is to authenticate such measurements.

This motivates the novel concept of PAtt: sensor readings from the physical process are used to authenticate the attestation response. The software attestation result is tied to the physical process readings through a derivation of an actuation path from the cryptographic hash of the control logic. We now introduce the system and attacker model and then provide an overview of the proposed system.

3.1 System and Attacker Model

The industrial control system considered in this work consists of a PLC that is controlling a dynamic local physical process such as a robotic arm, a laser, or extruder. The control logic of the PLC is responsible for real-time sensing and actuation of the dynamic process, e.g., periodic transport of a manufacturing component from one position to another position. The PLC does not provide onboard support for trusted execution or cryptographic signatures. Instead, the PLC does provide the capability to compute cryptographic hash functions over one or more data blocks used by the control logic (e.g., as possible on the Siemens S7 series PLCs). A remote attestation server (the "verifier") is connected to the PLC over the local network and is attempting to verify the correct state of the system. Attacks that compromise the attestation server are out of the scope of this work. The verifier has a model of the physical processes that are trained during normal operation in the absence of the attacker.

We consider an adversary that has compromised the PLC. The attacker's goal is to change the way the physical process is actuated while hiding this compromise from the attestation server. The attacker is limited to executing code on the compromised PLC, which has limited computational power and memory. In particular, the attacker does not have additional computation devices inside the industrial network. As the attacker has compromised the PLC, attacks that would manipulate the PLC firmware is subsumed in our attacker model (as the data could also be manipulated by the PLC firmware when sent or received). We considered two types of attackers:

1. Hash approximation: This attack is designed for evaluating decoding precision. In this attack, a number hash bits will be flipped at a random offset of the hash.

2. Replay attack: In this attack, the attacker will replay a stored sensor reading inside the PLC that corresponds to a subset of the actual hash.

3.2 PAtt Framework

We now propose the Physics-based attestation (PAtt) framework, which allows the attestation server (AS) to perform remote attestation of the PLC's currently loaded control logic.

Overview. The main steps of PAtt are summarized in Figure 1: 1) the AS initiates the attestation process by sending a fresh challenge nonce over the network using industrial protocols; 2) the PLC stores this nonce as data block in the memory accessible to the logic code, and the PLC then computes a cryptographic hash over the PLC logic code blocks including the nonce; 3) the resulting hash is interpreted as an actuation strategy for the robotic arm (details on this are provided later) and the movement path is executed by the actuator; 4) the resulting sensor readings during this movement are collected (sent to the AS together with the hash), and the AS verifies

that the hash was derived from correct PLC logic and the nonce, and that the sensor measurements fit the specific physical process and hash. If the last step is successful, the AS has attested the integrity of the logic on the PLC. In practice, only a limited number of actuation can be executed within a scan cycle of a PLC (e.g., 10ms), which might require us to run multiple iterations (or rounds) of the protocol. We defer discussion of that implementation detail to Section 4. We now provide additional details on each step.

1) Attestation Request. The AS initiates the attestation request by sending the PLC a nonce (a randomly generated bit vector) using the standard industrial protocols. The PLC firmware receives the nonce and makes it available to the PLC control logic as normal data tag. Figure 2 shows a more detailed view of the interactions between the verifier, PLC, and the physical process during the attestation process.

2) Nonce Storage and Hash Computation. The PLC then computes a cryptographic hash function over a group of control logic objects (standard blocks, safety blocks, text lists, and the received nonce from the verifier). If the hash function is also able to cover the firmware memory space, that region should be included in the hash as well. We note that in our implementation, the specific PLC used is only able to cover logic accessible memory areas, and thus not the RTOS.

3) Derivation of Actuation Strategy, and Execution of Path. The cryptographic hash generated in the previous step is then encoded to an actuation strategy for the actuator controlled by the PLC. The intuition is that the execution of such strategies will (deterministically) create sensor readings unique for the physical process. To not impede on normal process operations, the actuation strategy should essentially represent an alternative way to reaching the normal operational goal of the process, without violating safety constraints. Additional details on the derivation of the actuation strategy are provided in the following subsections.

4) Verification of Hash and Measurements. Each physical process is unique, which will be reflected in a process-specific noise signal in all reported sensor values. In PAtt, the verifier knows the unique noise signature of all prover devices and can, therefore, identify all PLCs and the connected physical systems based on the sensor readings transmitted during attestation, i.e., the attested PLC is authenticated in the process through the reported sensor values. This noise includes any source of random noise e.g., the noise of the system, manufacturing imperfections, and differences. This ensures that the reported sensor values authenticate the attestation report. In Section 3.4, we propose to use machine learning techniques to compute the classifier prediction probability. Along with decoding the hash, we use the classifier prediction probability as a weight to compute the weighted distance between the reconstructed hash and the original hash. We describe the implementation details of the hash verification in Section 4.

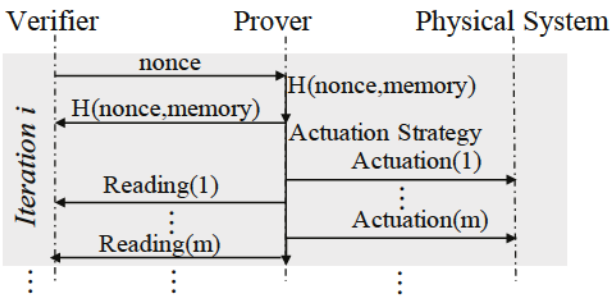


Figure 2: Interaction sequence of the verifier, the prover, and the physical system during attestation.

3.3 Generation of Actuation Strategy

We now discuss how the hash value of the logic-accessible memory areas and the nonce can be interpreted as an actuation strategy. In general, we assume that the cryptographic hash has length m bits. Assuming that there are two different potential actuation actions (e.g., a move horizontally or vertically), we interpret the hash as a sequence of those binary actuation commands, with m commands being executed per iteration. Figure 2 shows the interaction sequences of the verifier, PLC, and the physical process during the remote attestation process. We now provide additional details on individual commands in the strategy (which we call micro-commands).

Macro-commands and Micro-commands. Macro-commands are abstract movements to reach a goal from the start. The macro-command can be executed by different sequences of micro-commands (called a path strategy). Consider the following example: a robot arm with three stepper motors, which when actuated together will change the arm hand position in the x, or y, or z directions. The designed control logic of the robot arm will traverse the arm hand from the position (x_0, y_0, z_0) to the position (x_1, y_1, z_1). We define the macro-command as move from (x_0, y_0, z_0) to the position (x_1, y_1, z_1). The coordinate system is scaled such that each micro-command moves the arm by one unit (in particular, the arm does not move diagonally).

Path Strategy. Continuing our example, the path strategy now determines the sequence of micro-commands to execute the macro-command. To simplify things, we only consider micro-commands in direction x and y (and always use the same z direction commands). The arm hand will start from (x_0, y_0, z_0) and takes $x_1 - x_0$ steps towards the x direction, and $y_1 - y_0$ steps towards the y direction. Thus, the total number of steps (or micro-commands) is $(x_1 - x_0) + (y_1 - y_0)$. The order of micro-commands is defined by the path strategy.

We represent the path strategy as a binary vector (which we call a coding), with a micro-command in x-axis direction represented by a 0 bit, a micro-command in y-axis direction represented by a 1 bit. For example, our robot arm goal

might be to take five steps in the x direction and to take three steps in the y direction. Two possible path strategies could be `10100010`, and `00000111`. Figure 8 in Appendices 10.4 shows an example path strategy. We know that the number of unique paths u in a $x \times y$ grid can be computed as follows [57]:

$$u = \frac{(x+y)!}{(x!y!)} \tag{1}$$

Thus, we can enumerate all possible paths, and use an integer between 1 and u as an index to represent a specific path strategy in a $x \times y$ grid. In PAtt, this index is a random number resulting from the software attestation phase (i.e., the resulting cryptographic hash is interpreted as integer). The micro-commands of the chosen actuation strategy is then executed, and sensor readings are recorded to be sent as part of the attestation response.

3.4 Verification of the Measurement Traces

In the verification phase, the Verifier checks that the hash received from the Prover was derived from correct PLC logic and the nonce, and that the sensor measurements fit the specific physical process and hash. If the last step is successful, the Verifier has attested the integrity of the logic on the PLC.

Hash verification. As the Verifier has access to the logic that is supposed to run on the PLC (and the nonce), it is easy to check if the hash is correct. In particular, the Verifier computes the hash locally and compares with the received hash.

Replay Attack Detection. Next, the Verifier has to ensure that the received sensor reading sequence fits the received hash (which we call decoding) and that the sensor reading sequence was not spoofed (e.g., by simulation of the physical process or replay attack). The decoding process is designed to translate the sensor readings to the original hashes considering the physical behavior of the system and its non-deterministic noise. The features available are the actual sensor reading traces and information on the physical process (e.g., statistical properties of noise from the sensors) that were measured during the setup phase of the system. The decoding can be done by signal processing techniques (e.g., matched filters that detect movements and reconstruct the actuation strategy/hash), or machine learning approaches. The detection of replayed sensor traces can similarly be performed by statistical analysis, signal processing techniques, or machine learning.

3.5 Security Analysis

As the hash received is a result of weak software-based attestation, the verifier also needs to check if the hash was received within a specific time window. In particular, we need to prevent a compromised PLC from sharing the received challenge with a third party oracle that would provide the correct hash. As PLCs are only able to send and received messages synchronized with scan cycles, we use two scan cycles as a maximal

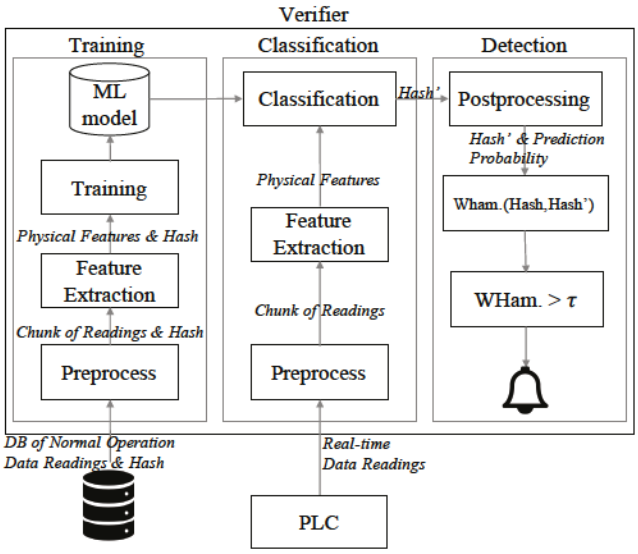


Figure 3: Overview of the validation process in the Verifier.

delay to provide the hash (in the first scan cycle, a challenge is received, and in the second the hash is computed and sent to the Verifier). In our implementation, scan cycles are 10ms long, so the maximal delay to provide the hash is 20ms.

Full replays of earlier actuation sequences are not feasible for the attacker, as she would need to eavesdrop and store a significant share of the hash space (i.e., actuation space and its corresponding sensor readings) in order to reliably be able to perform the replay attack. As the hash contains a fresh nonce, and the hash itself has 256-bit length, we consider this infeasible. As an example, a SIEMENS S7-1200 PLC might have work memory up to 125 Kilobytes, and load memory up to 4 Megabytes. A single trace of sensor readings of a single SHA-256 hash will require more than 300 Kilobytes, hence, storing the possible hashes inside the PLC is infeasible. As we will show in Section 5, in practice our decoding solution was able to detect attacks that could be produced on constrained devices such as PLCs, so there we are able to do both decoding and spoofing detection in one step.

4 Implementation

In this section, we provide details on our practical implementation of PAtt in an ICS use-case. We start by presenting our solution for hash verification and spoofing detection. Then, we present the ICS setup in which we evaluated PAtt.

4.1 Machine Learning Decoder

In our implementation, we chose to use (supervised) machine learning based approaches to both decode the sensor reading traces to the hash and detect spoofing of sensor reading traces. In Figure 3, we provide an overview

of the data processing at the verifier, including the offline training phase. The classifier decodes each individual micro-command from the sensor reading trace to its corresponding bit (or step) in the actuation strategy. To select an appropriate classifier, we implemented and evaluated a number of classifiers using WEKA [23]. In particular, we tested Random Forests [11], Multilayer Perceptrons [7], Decision Forests [25], FURIA [26], DTNB [24] NBTree [31], LMT [34], J48 [48], PART [19], and REPTree [63]. To benchmark the machine learning model used in our verifier, we have evaluated a set of classifiers that are mostly used in related security research works e.g. [4,5,45,51,56] as shown in Appendix 5.2. Now we thoroughly discuss different parts of the PAtt as presented in Figure 3.

1) Training. The features that we used are trajectory position of the arm, average movement over a window of time, and statistical features that are mostly used in the signal processing to monitor the signal behavior. These statistical features are the mean of the signal over a window of time and the standard deviation of the signal. We applied the preprocessing of the signals that were reported by the sensor, and we use those features to recover the hash with a probability of prediction of the classifiers. We used the current position, the mean (AVG) and standard deviation (STD) of the Accelerometer and Gyroscope, and the change of mean of Accelerometer as features to create the profile of the physical system (see Table 1). Considering the three-dimensional trajectory of the robotic arm and two Accelerometers and two Gyroscopes, we used 33 features to train the machine learning model, and SciPy and NumPy libraries of Python [12] to automatically generate these features. In total, feature extraction was roughly 300 lines of code.

Table 1: The features classes used in PAtt.

Feature	Formula
Current Position	(x_A, y_A, z_A)
Mean of Accelometer	$\text{AVG}_t(x_A, y_A, z_A)$
STD of Accelometer	$\text{STD}_t(x_A, y_A, z_A)$
Accelometer Difference	$\text{AVG}_t - \text{AVG}_{t-1}$
Mean of Gyroscope	$\text{AVG}_t(x_G, y_G, z_G)$
STD of Gyroscope	$\text{STD}_t(x_G, y_G, z_G)$

2) Classification. The classifiers assign a class to the test data set with a probabilistic model. Considering the sample set X, and class labels form a finite set Y, the classifier would assign a conditional distribution $\Pr(Y|X)$ which for a given sample $x \in X$, the classifier would assign a probability of being in $y \in Y$ class. Depending on the classification method, in hard classification, the sample $x \in X$ will be classified as $y \in Y$ class, where it holds:

$$\hat{y} = \arg\max_y \Pr(Y = y|X) \tag{2}$$

We use the highest classifier prediction probability of decoding a bit as a weight in the weighted distance computation.

3) Detection. In PAtt, the Prover generates a hash from the random nonce and memory block. Then it creates the actuation strategy based on the derived hash, and the physical system performs the actuation strategy and reports the sensor reading traces to the Prover. After decoding this hash (described above), the Verifier needs to decide whether it is authentic, for which we propose to use a weighted Hamming distance. To compute the weighted distance of the original hash and the recovered hash, we used the weighted Hamming distance with the classifier prediction probability as a weight of each bit of the hash. The Hamming distance is the number of non-matching positions between two equal-length string. Considering a noisy channel of transmitting bit arrays, the Hamming distance could be used to determine how many bits are different from the original bit arrays. In this paper, we used the Hamming distance to calculate the distance between the original hash and the decoded hash transmitted by actuation commands to the physical process and retrieved from the physical process by sensor readings. The Hamming distance between two-bit arrays of $a[1..k]$ and $b[1..k]$ is computed by:

$$\text{Ham}(a,b) = \sum_{i=1}^{k} a_i \oplus b_i \tag{3}$$

We use the weight of specific bits in the distance computation of the Hamming distance. The PAtt is using the classifier prediction probability as the weight for each bit of the hash. The weighted Hamming distance between two bit arrays of $a[1..k]$ and $b[1..k]$, and weight $w[1..k]$ is computed by:

$$\text{WHam}(a,b) = \sum_{i=1}^{k} w_i(a_i \oplus b_i) \tag{4}$$

The computed weighted Hamming distance over original hash and the recovered hash will be:

$$\text{WHam}(Hash, Hash') = \sum_{i=1}^{k} CPr_i(Hash_i \oplus Hash'_i) \tag{5}$$

where *Hash* is the original hash, *Hash'* is the recovered hash, i is the index of the bit, and CPr_i is the classifier prediction probability of the i^{th} bit. PAtt will trigger an alarm when the computed weighted Hamming distance passes the detection threshold (τ).

4.2 Testbed Design and Setup

In this section, we describe our industrial robot arm testbed and our implementation of PAtt's Prover on the PLC controlling the robot arm.

The PLC and Process. The industrial control system used in this paper is a modern robotic arm controlled by a Siemens

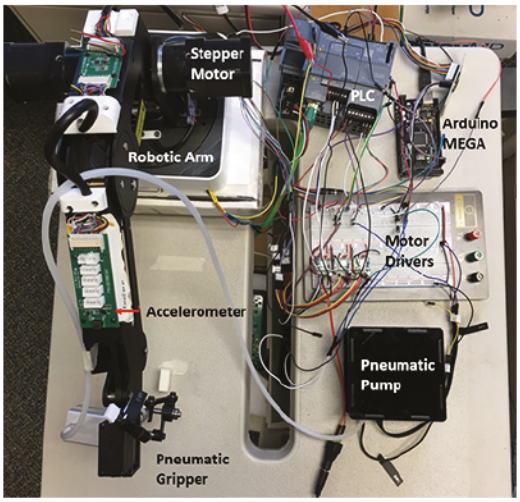


Figure 4: Implemented setup.

S7-1200 PLC with a remote attestation server with at least one GB storage, 16 GB memory, and an Intel Core i7 processor. The PLC is programmed with the controller code for the robotic arm. The verifier communicates with the PLC over an industrial network and uses the SNAP7 library. The verifier is written in Python, and has more than 3000 lines of code.

Our setup consists of a PLC-controlled Dobot robotic arm. The arm includes a rotating base with two arm segments, which we refer to as the rear and front arm segments respectively. At the end of the front arm, the segment is an end-effector, which can hold several different attachments, like a gripper or 3D-printing extruder head. The arm is actuated by three stepper motors, which we control via a Siemens S7-1200 PLC. Actuation is achieved through (i) a Pulse Width Modulation (PWM) Input/Output (IO) module for the PLC, and (ii) one A4988 stepper driver per stepper motor to correctly control the motor coils. Due to limitations in the amount of IO pins available in our setup, we utilize only two stepper motors in our implementation (see Figure 4). The PLC controls the stepper drivers with PWM signals generated by our designed function blocks of the PLC control logic that translate the macro-commands to micro-commands, including the analog conversion of the hash.

For sensing, the robotic arm uses an inertial measurement unit (IMU) on both of its segments, with accelerometer and gyroscope measurements collected during operation being routed to the PLC via a combination of an Arduino MEGA and MAX232 adapter, converting the accelerometers' I2C protocol to an RS232 module on the PLC.

On the PLC, we program the sensing-actuation control loop. We implement the low-level arm kinematics, and point-to-point movement functions adapted from the open-dobot project [1] translated to the Siemens SCL language. To simulate an application-focused environment, we use python scripts to allow for automated control and link them to the PLC through the Snap7 communication library.

Function Blocks for Prover Functionality. To simplify the usage of the proposed method, in practical industrial control systems we programmed the whole path strategy and attestation functions inside the function blocks that could be easily used by non-experts. The core idea of the PAtt is to enable the remote attestation of a PLC's control logic (or the configuration data) without requiring changes to the PLC hardware or the manufacturer provided firmware. In other words, PAtt should be applicable to legacy systems by re-compiling/extending only the operator provided control logic programs of a PLC. We wrote 315 lines of code inside the designed function blocks to compute the hash and perform the actuation strategies. We used TIA 15 together with the Python-Snap7 libraries at the verifier side to communicate with the function blocks inside the PLC.

The attestation result, as soon as it is computed, will be encoded into the commands the PLC sends to the connected physical system as well as the verifier. The resulting sensor readings that depend on the attestation report are forwarded to the verifier. The verifier has a model of the physical system and can calculate which series of sensor readings to expect based on the actual functional commands, including the encoded attestation report.

Random Actuation Strategy. To implement the required random actuation strategy, we designed a set of function blocks to perform PAtt's micro-commands inside the function block within the bounded timing of the needed abstract macro-command by the process. As discussed before, the completion of the micro-commands will lead to a physical state as required by the macro-command. The random actuation strategy was written directly inside the function blocks, and it includes three function blocks and in total, 214 lines of codes inside the ladder logic. As only a limited number of micro-commands can be executed in each scan cycle, we are not able to run the complete actuation strategy based (with a number of steps determined by the length of our hash) within a single scan cycle. In particular, in our implementation, we generate eight micro-commands in one scan cycle. We now discuss how we addressed this implementation issue.

Splitting the Attestation over Multiple Protocol Rounds. Overall, we want to execute 256 micro-commands (to match the 256-bit output of our hash, SHA256). As we can only execute eight micro-commands in each scan cycle, we need to execute micro-commands over multiple scan cycles. Unfortunately, this increases the time required for the attestation, and potentially allows the attacker more time to compute precise simulations of executions (or communicate with a remote Oracle). To ensure that this is not possible for the attacker, we run the overall protocols in 32 rounds, with fresh nonces in each round. In each round, we execute eight micro-commands based on the most recent hash. Each round contributes 8 bits

of difficulty for the attacker to correctly guess the hash (or spoof sensor reading traces that are decoded to the expected hash). The Prover is attested by the Verifier only if all 32 rounds conclude successfully. We designed the encoding process to keep the PLC busy by the generation of hashes over the physical process and providing a new nonce in each scan cycle that convince the PLC to perform the random physical actuation over each scan cycle while the PLC computation power is devoted for the new hash generation.

5 Evaluation

In this section, we evaluate our proposal of the physics-based attestation of the control systems.

5.1 Dataset

Normal Operation. Our training dataset consists of 10 hashes and corresponding sensor reading traces, repeated six times for each hash (in total 60 runs). The traces are generated during normal operation (which captures the natural noise of the physical process) with a sampling rate of ten sensor readings for each micro-command. For our test evaluation dataset, we performed a number of attacks: a one-bit approximation attack, two-bit approximation attack, and simulation attacks (each attack was run 20 times). The overall size of sensor readings and the generated dataset was roughly 100 megabytes. We now describe the details of our attack implementation.

Hash Approximation Attacks. In this attack, an attacker modified the control logic of the PLC which resulted in a different hash (and thus a changed sequence of micro-commands). To show that PAtt can detect even minor changes in the hash/actuation sequence, we evaluated the performance of PAtt against an attacker that flips one or two random bits of the hash to modify the robot arm's trajectory. The attacker has full access to the stored hash and the actuation commands.

In particular, we performed ten one-bit approximation attacks and ten two-bit approximation attacks, and we evaluated the detection performance based on this attack. As the classifiers of our implementation PAtt are only trained on normal process behavior, we did not need to include the attack traces in the classifier training process. The implemented hash approximation attack was used as a test data-set, which consists of twenty hashes of sensor traces, and each hash repeated two times, and the normal operation was used as train data-set.

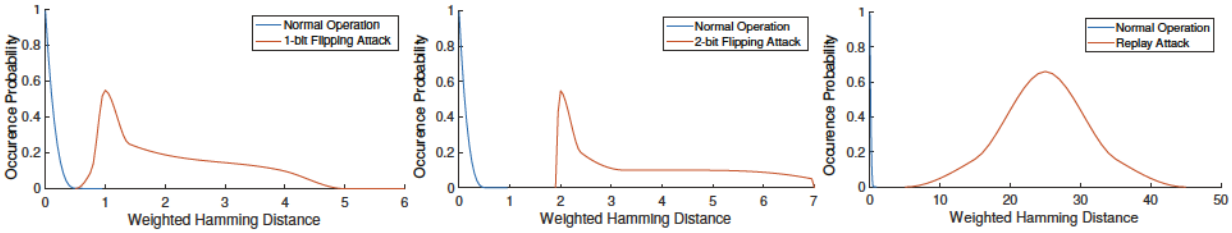
Replay Attacks. In addition to the effects of incorrect hashes or manipulated micro-command sequences, we also investigated whether the attacker would be able to produce sensor reading sequences by simulation (e.g., based on prior observations). For example, the attacker could try to record and replay earlier sensor reading traces, or try to re-assemble a new sensor reading trace from multiple earlier observations. We now argue why the former is infeasible and then describe how we evaluated the latter. The attacker could generate a table of short sequences of micro-commands and corresponding sensor readings to simulate sensor reading traces of arbitrary hashes. Our intuition is that this is not feasible as the sensor readings are influenced by the trajectory position of the arm and the average move over a window of time. Considering the memory and computational resources available on the PLC, the attacker needs to regenerate the corresponding sensor reading that is influenced by the trajectory position of the arm and the average move over a window of time. Our results (presented next) confirm this. The implemented replay attack was used as a test data-set which consists of ten hashes with replayed sensor traces from the attack table, and each hash repeated two times, and the normal operation was used as train data-set.

5.2 Evaluation Results

To evaluate the performance of PAtt, we use the following metrics (see Section 10.2 in the Appendix): Sensitivity, Specificity, Precision, False Positive Rate (FPR), False Negative Rate (FNR), Accuracy, F1-score, and Matthews Correlation Coefficient (MCC).

Decoding. We now present our evaluation of different classifiers for decoding the sensor reading traces to the actuation strategy. The results in Table 2 summarizes the performance of our classifiers. The most promising classifiers for our data set were the Random Forest and Multi-Layer Perceptron (with accuracy 0.9923 and 0.9915, respectively). The tradeoff between false acceptance and false rejection can be seen in the ROC, provided in Appendix 10.3. Overall, it can be observed that our decoding is reliably able to translate the sensor reading traces back into the micro-commands.

Hash Authentication. For remaining analysis, we used the RF classifier for the decoding. The next step (the hash authentication) required an appropriate value for τ (the threshold for the weighted Hamming distance between the expected and decoded hash). We now show how we selected τ, based on analysis of the distributions of the hashes' weighted Hamming distances in normal and attack cases. We start with (more intuitively understandable) figures on the distributions of weighted Hamming distances in normal operations and during attacks. Figure 5 shows the weighted Hamming distance with the occurrence probability distribution of one-bit approximation attack, two-bit approximation attack, and replay attack, respectively. As shown in Figure 5a, the curve of the occurrence probability distribution of the normal operation overlap with the curve of the occurrence probability distribution of the one-bit approximation attack. That implies no value for τ will allow us to decide between the two cases without error correctly, and it means that there will be a non-negligible probability that the attacker can perform her attack while remaining undetected. In contrast, Figure 5b shows the curve of the occurrence probability distribution of the normal



(a) Comparison between the weighted Hamming distance of normal operation and 1-bit hash approximation attack.

(b) Comparison between the weighted Hamming distance of normal operation and 2-bit hash approximation attack.

(c) Comparison between the weighted Hamming distance of normal operation and replay attack.

Figure 5: Weighted Hamming distance against the attack use-cases.

Table 2: Performance comparison of different decoding classifiers sorted by accuracy. FPR=False Positive Rate, FNR=False Negative Rate, MCC=Matthews Correlation Coefficient.

Algorithms	Sensitivity	Specificity	Precision	FPR	FNR	Accuracy	F1-score	MCC
Random Forest	0.9926	0.9921	0.9920	0.0079	0.0074	0.9923	0.9923	0.9847
Multilayer Perceptron	0.9915	0.9916	0.9915	0.0084	0.0085	0.9915	0.9915	0.9831
Decision Forest	0.9915	0.9895	0.9894	0.0105	0.0085	0.9905	0.9904	0.9810
FURIA	0.9878	0.9921	0.9920	0.0079	0.0122	0.9899	0.9899	0.9799
DTNB	0.9857	0.9910	0.9910	0.0090	0.0143	0.9884	0.9884	0.9767
NBTree	0.9873	0.9889	0.9889	0.0111	0.0127	0.9881	0.9881	0.9762
LMT	0.9873	0.9879	0.9878	0.0121	0.0127	0.9876	0.9875	0.9751
J48	0.9867	0.9868	0.9867	0.0132	0.0133	0.9868	0.9867	0.9735
PART	0.9893	0.9832	0.9830	0.0168	0.0107	0.9862	0.9862	0.9725
REPTree	0.9819	0.9810	0.9809	0.0190	0.0181	0.9815	0.9814	0.9630

operation does not overlap with the curve of the occurrence probability distribution of the two-bit approximation attack, which demonstrates that we can precisely detect the two-bit approximation attack (e.g., by choosing $\tau = 1$). Figure 5c shows the curve of the occurrence probability distribution of the normal operation did not overlap with the curve of the occurrence probability distribution of the replay attack, which means that we can also precisely detect replay attacks (e.g., by choosing $\tau = 1$).

In our experiments (see Table 3), first we studied the performance of PAtt with different values of the τ, from 0.8 to 1. We used the 60 traces of the normal operation and 60 traces of attacks in total. The true negative (TN) is the number of normal operation instances that are correctly classified as normal operation. The false positive (FP) is the number of normal operation instances that are wrongly classified as an attack. The true positive (TP) is the number of attack instances that are correctly classified as an attack. The false-negative (FN) is the number of attack instances that are wrongly classified as normal operation. We confirmed that $\tau = 1$ yields ideal performance in normal operations of the system. In the absence of attacks, our processing of the sensor reading traces always produces a hash that is classified as authentic. In addition, we can detect the implemented attacks attack with Accuracy of 89%, sensitivity of 78%, and Matthews Correlation Coefficient (MCC) of 0.80. Both the two hash approximation attack and replay attack was detected without any false negative. Choosing the best value of the τ is dependent on the operational requirements of the control processes. However, if the operation of the system could tolerate the false positives (which would probably trigger the alarm even during the normal operation), we could choose the $\tau = 0.95$ which can detect the implemented attacks attack with Accuracy of 97%, the sensitivity of 96%, and MCC of 0.95.

6 Discussion

We now provide an additional discussion on the scalability of our approach, practical issues with critical zones, and the use-case scenarios.

6.1 Complexity/Scalability

The PAtt uses physical complexity (physical behavior) as a root-of-trust. By adding more sensors or actuators to the control processes, we could achieve a more robust model that

Table 3: Attack Detection performance comparison of implemented attacks (RF-based decoding). TN=True Negative, FP=False Positive, TP=True Positive, FN=False Negative, MCC=Matthews Correlation Coefficient.

Threshold	Normal		1-bit		2-bit		Replay		Sensitivity	Accuracy	F1-score	MCC
	TN	FP	TP	FN	TP	FN	TP	FN				
$\tau = 1$	60	0	7	13	20	0	20	0	0.7833	0.8917	0.8785	0.8024
$\tau = 0.95$	59	1	18	2	20	0	20	0	0.9667	0.9750	0.9748	0.9501
$\tau = 0.9$	58	2	18	2	20	0	20	0	0.9667	0.9667	0.9667	0.9333
$\tau = 0.8$	54	6	19	1	20	0	20	0	0.9833	0.9417	0.9440	0.8864

could verify the integrity of the control processes by wrapping this complexity over actuation strategy and the hash. In addition to the favor of complexity, the PAtt is designed to be scalable, and it could detect the change of physical complexity (physical behavior) as we have seen in the Section 5 by detecting the replay attacks which would report a table-based replay of physical behavior. These features of the PAtt would make it feasible to authenticate a physical process over an actuation strategy derived from a random hash.

6.2 Application to other PLCs

We used some APIs from the S7-1200 PLCs to perform the memory measurements and hash generation. The same functionality can be provided on other PLCs if they support the APIs. However, the memory measurement could be programmed directly in the control logic of the PLCs with some engineering effort.

6.3 Critical Zones

Given that this attestation routine is being integrated into safety-critical processes, there are restrictions on when the attestation process can be performed. We refer to these restrictions as critical zones. During a critical zone, the physical process is engaged in a fixed actuation and thus cannot be interrupted by an attestation. As an example, in 3D printing, a critical zone would be when the printer head is extruding filament. In addition to timing, the safety-critical zone must also include a spacial component, as actuation generated by the attestation process must not collide with anything and also stay within the range of motion of the system. A consequence of this is that processes that have no downtime or are always performing some critical action cannot utilize this augmented attestation method.

6.4 Example Applicable Use-Case Scenarios

In this section, we describe the application of our attestation scheme in several use-cases. We designed and implemented PAtt in a robotic arm controlled by a PLC, and we showed the applicability and security significance of the PAtt in a real-world ICS. However, PAtt applies to other CPS as well, where the control process meets the following conditions:

- The control process has the ability to perform high-speed actuation (such as the 200 kHz PWM signal board that we used in the implementation of PAtt).
- The control process has powerful sensors that are able to report the current state of the physical process via a high-speed channel.

Automated Manufacturing. The first use case is automated manufacturing, which involves machinery similar to our implementation on a robotic arm. These types of setups are common in automotive assembly, where a robotic arm manipulates objects in 3D space. Actuation is carefully controlled and monitored by a PLC. In this situation, the critical zone occurs when the arm is manipulating an object. Conversely, when the arm is not gripping an object, the attestation process can be initiated. As described in the previous section, the conventional attestation report is encoded into micro-actuation of the robotic arm, resulting in sensor readings to corroborate the authenticity of the attestation report. In this case, the initiation of the attestation process must also take into account the spacial constraints given that the arm must not contact any objects during the attestation process.

Additive and Subtractive Manufacturing. Additive manufacturing processes, most commonly referring to 3D printing, consist of a printer head extruding heated filament in successive 2D layers to produce a product. The printing process is controlled by a micro-controller controlling several stepper motors. Designs are created in one of the various 3D printing programs and converted to a standard language of instructions known as G-code. In recent years, this technology has seen rapid growth leading not only to 3D printed parts used in a greater variety of applications, including safety-critical ones like medical prostheses. Consequently, it is essential to consider the security aspect of these applications, and much work has been done in this area already. The application of our attestation scheme to this process is complicated by the fact that conventional 3D printers lack the physical sensor channels leveraged in the robotic arm use-case, only being equipped with several limit switches for initial calibration,

then determining position relative to a "home point" through hard-coded characterization of the stepper motors controlling the motion of the printer head. However, this can easily be remedied through the placement of external sensors, like the accelerometers used in the robotic arm case, on to the printer head to provide a physical sensing channel to provide actuation feedback for the attestation procedure. In this use-case, the critical zone takes place when the printer head is extruding filament; its path is fixed. Thus the attestation procedure can take place between printing layers after the printer head is raised to satisfy the spacial aspect of the critical zone. A similar argument can be made for subtractive manufacturing applications, like laser engraving or CNC milling. Their setups are similar to 3D printing, with precise but relative motion control and an on/off-type process dictating their critical zones. Once again, the addition of cost-effective external sensors can allow the attestation of these processes.

7 Related Work

Remote Attestation. Remote attestation for embedded systems has been studied form many years, however, typically these approaches do not provide real-time execution guarantees for the system (at least while attestation is in progress) [10, 15, 30, 59]. The fundamental conflict between real-time execution and attestation has to be discussed by Carpent et al. [13]. Additionally, these approaches rely on a (hardware) root-of-trust, which is not available in today's CPS. Software-based attestation does not require a root-of-trust [28, 36, 37, 52–54], but can only provide uncertain security guarantees [55]. Valente et al. propose to leverage control-command variations to validate the correctness of a system's physical operations rather than the system software integrity, providing "a weak attestation" [61]. Our proposed scheme combines software-based attestation with control-command mutations to overcome the limitations of software-based attestation while providing guarantees regarding a device's software-integrity.

Machine Learning based Anomaly Detection. The authors of [42] discussed machine learning proposals for anomaly detection in the ICS. Also, the authors of [33] proposed to use convolutional neural networks for detecting cyber attacks on industrial control systems. Machine learning techniques for anomaly detection in industrial arm applications is discussed in [43]. Decision trees used in several security research areas like anomaly detection in network traffic data [8]. The authors of [44] used deep learning architecture in correlating Tor connections. The authors of [39] used the k-means clustering to detect traffic phase shifts inside the SCADA automatically. In [16], the authors discussed data mining and machine learning techniques for cybersecurity. There are many successful applications of machine learning in cyber-physical system security. PAtt uses the machine learning to learn the physical behavior alongside the other security measures like the integrity and authenticity checking of the control logic of the PLCs. Also, PAtt do not need to see the attack during the training phase. PAtt do not need to see the attack during the training phase, and it could detect the attack by a distance metric. Also, unlike most of machine learning anomaly detection techniques, PAtt is able to identify the replay attack.

Physical Unclonable Functions. The authors of [62] proposed to use the MEMS gyroscope as physical unclonable function, and they showed the feasibility of using the physical and electrical properties of the MEMS gyroscope for cryptographic key generation. The authors of [35] proposed an attested execution processor that do not need secure non-volatile memory, and it derives cryptographic identities from manufacturing variation measured by a PUF. Aging effects in PUFs have been discussed in the [38, 46], which were dominated by physical effects in the resistors. We evaluated the aging effect on industrial actuators and sensors used in our experimental evaluation in the Appendix 10.1, and show that on the scale of our sensors, no aging effects were observed.

8 Conclusion

In this paper, we presented PAtt, a novel remote attestation scheme for control processes of industrial control systems that integrates software-based attestation with physical behavior correlations, similar to a PUF. We used software-based attestation techniques to first generate a fresh hash over the loaded logic in the PLC, which is then translated into an actuation strategy for the physical process. The resulting sensor reading traces are then checked by the verifier to ensure that the correct logic is loaded and to detect spoofing of the process data. The proposed solution was able to detect the attack that is not seen before during the training phase of the verifier, and it could measure the anomalies by computing the distance over a cryptographic hash generated from the attestation of the control process. We implemented our solution (based on a robotic arm test-bed with Siemens PLC), and show that our proposed solution is accurate enough to detect the tested attacks with an accuracy of 97%, sensitivity of 96%, and Matthews Correlation Coefficient of 0.95 ($\tau = 0.95$).

9 Acknowledgment

The authors would like to thank the Singapore University of Technology and Design (SUTD), TU Darmstadt, DAAD, Rutgers University, CISPA- Helmholtz Center for Information Security, and UCLA for supporting this research by providing financial means and access to the laboratories. This work has been supported by the German Research Foundation (DFG) as part of projects HWSec, P3 and S2 within the CRC 1119 CROSSING, by the German Federal Ministry of Education and Research (BMBF) and the Hessen State Min-

istry for Higher Education, Research and the Arts (HMWK) within CRISP, by BMBF within the projects iBlockchain and CloudProtect, and by the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS). Jianying Zhou's work is supported by the National Research Foundation (NRF), Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2014NCR-NCR001-31) and administered by the National Cybersecurity R&D Directorate. This research is also funded in part by the National Science Foundation under awards CNS-1703782 and CNS-1705135. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, or the U.S. Government. We thank the National Science Foundation (NSF) - Cyber-Physical Systems (CPS) program - for their support of this project. Additionally, This material is partially based upon work supported by the Department of Energy's Office of Cybersecurity, Energy Security, and Emergency Response and the Department of Homeland Security's Security Science & Technology Directorate under Award Number DE-OE0000780.

References

[1] open-dobot. https://github.com/maxosprojects/open-dobot. Accessed: 2018-12-06.

[2] D. Antonioli, H. R. Ghaeini, S. Adepu, M. Ochoa, and N. O. Tippenhauer. Gamifying ics security training and research: Design, implementation, and results of s3. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 93–102. ACM, 2017.

[3] F. Armknecht, A.-R. Sadeghi, S. Schulz, and C. Wachsmann. A security framework for the analysis and design of software attestation. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1–12, New York, NY, USA, 2013. ACM.

[4] S. Banescu, C. Collberg, and A. Pretschner. Predicting the resilience of obfuscated code against symbolic execution attacks via machine learning. In *Proceedings of the 26th USENIX Security Symposium*, 2017.

[5] D. Barradas, N. Santos, and L. Rodrigues. Effective detection of multimedia protocol tunneling using machine learning. In *USENIX Security*, 2018.

[6] Z. Basnight, J. Butts, J. Lopez, and T. Dube. Firmware modification attacks on programmable logic controllers. *International Journal of Critical Infrastructure Protection*, 6(2):76–84, 2013.

[7] E. B. Baum. On the capabilities of multilayer perceptrons. *Journal of complexity*, 4(3):193–215, 1988.

[8] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: methods, systems and tools. *IEEE communications surveys & tutorials*, 16(1):303–336, 2014.

[9] W. Bolton. *Programmable logic controllers*. Newnes, 2015.

[10] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl. TyTAN: Tiny Trust Anchor for Tiny Devices. In *ACM DAC*, 2015.

[11] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[12] E. Bressert. *SciPy and NumPy: an overview for developers*. " O'Reilly Media, Inc.", 2012.

[13] X. Carpent, K. Eldefrawy, N. Rattanavipanon, A.-R. Sadeghi, and G. Tsudik. Reconciling remote attestation and safety-critical operation on simple iot devices. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[14] E. Chien, L. OMurchu, and N. Falliere. W32.Duqu - The precursor to the next Stuxnet. Technical report, Symantic Security Response, nov 2011.

[15] K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik. SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust. In *NDSS*, 2012.

[16] S. Dua and X. Du. *Data mining and machine learning in cybersecurity*. CRC press, 2016.

[17] F-Secure Labs. BLACKENERGY and QUEDAGH: The convergence of crimeware and APT attacks, 2016.

[18] N. Falliere, L. O. Murchu, and E. Chien. W32.Stuxnet Dossier. Technical report, Symantic Security Response, Oct. 2010.

[19] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. 1998.

[20] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz. Hey, my malware knows physics! attacking plcs with physical model aware rootkit. In *Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA*, pages 26–28, 2017.

[21] H. R. Ghaeini, D. Antonioli, F. Brasser, A.-R. Sadeghi, and N. O. Tippenhauer. State-aware anomaly detection for industrial control systems. In *The 33rd ACM/SIGAPP Symposium On Applied Computing (SAC)*, Apr. 2018.

[22] H. R. Ghaeini and N. O. Tippenhauer. Hamids: Hierarchical monitoring intrusion detection system for industrial control systems. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 103–111. ACM, 2016.

[23] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[24] M. A. Hall and E. Frank. Combining naive bayes and decision tables. In *FLAIRS conference*, volume 2118, pages 318–319, 2008.

[25] T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[26] J. Hühn and E. Hüllermeier. Furia: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319, 2009.

[27] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann. PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (PUFs) cast in silicon. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 283–301, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[28] C. Kil, E. C. Sezer, A. M. Azab, P. Ning, and X. Zhang. Remote Attestation to Dynamic System Properties: Towards Providing Complete System Integrity Evidence. In *IEEE/IFIP DSN*, 2009.

[29] E. D. Knapp and J. T. Langill. *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress, 2014.

[30] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan. TrustLite: A Security Architecture for Tiny Embedded Devices. In *ACM EuroSys*, 2014.

[31] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Second International Conference on Knoledge Discovery and Data Mining*, pages 202–207, 1996.

[32] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann. PUFatt: Embedded platform attestation based on novel processor-based PUFs. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, pages 109:1–109:6, New York, NY, USA, 2014. ACM.

[33] M. Kravchik and A. Shabtai. Detecting cyber attacks in industrial control systems using convolutional neural networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 72–83. ACM, 2018.

[34] N. Landwehr, M. Hall, and E. Frank. Logistic model trees. *Machine learning*, 59(1-2):161–205, 2005.

[35] I. Lebedev, K. Hogan, and S. Devadas. Secure boot and remote attestation in the sanctum processor. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 46–60. IEEE, 2018.

[36] Y. Li, J. M. McCune, and A. Perrig. SBAP: Software-Based Attestation for Peripherals. In *TRUST*. Springer, 2010.

[37] Y. Li, J. M. McCune, and A. Perrig. VIPER: Verifying the Integrity of PERipherals Firmware. In *ACM CCS*, 2011.

[38] C. Q. Liu, Y. Cao, and C. H. Chang. Acro-puf: A low-power, reliable and aging-resilient current starved inverter-based ring oscillator physical unclonable function. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(12):3138–3149, 2017.

[39] C. Markman, A. Wool, and A. A. Cardenas. Temporal phase shifts in scada networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 84–89. ACM, 2018.

[40] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A.-R. Sadeghi, M. Maniatakos, and R. Karri. The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, 104(5):1039–1057, 2016.

[41] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel. A trusted safety verifier for process controller code. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2014.

[42] A. Meshram and C. Haas. Anomaly detection in industrial networks using machine learning: a roadmap. In *Machine Learning for Cyber Physical Systems*, pages 65–72. Springer, 2017.

[43] 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*, pages 13–23. ACM, 2018.

[44] M. Nasr, A. Bahramali, and A. Houmansadr. Deepcorr: Strong flow correlation attacks on tor using deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1962–1976. ACM, 2018.

[45] X. Pan, Y. Cao, X. Du, G. Fang, R. Shao, and Y. Chen. Flowcog: Context-aware semantics extraction and analysis of information flow leaks in android apps. In *USENIX Security*, 2017.

[46] M. T. Rahman, F. Rahman, D. Forte, and M. Tehranipoor. An aging-resistant ro-puf for reliable key generation. *IEEE Transactions on Emerging Topics in Computing*, 4(3):335–348, 2016.

[47] J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell. A quantitative evaluation of the target selection of havex ics malware plugin. In *Industrial Control System Security (ICSS) Workshop*, 2015.

[48] S. Ruggieri. Efficient c4. 5 [classification algorithm]. *IEEE transactions on knowledge and data engineering*, 14(2):438–444, 2002.

[49] A.-R. Sadeghi and C. Stüble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. In *Proceedings of the 2004 Workshop on New Security Paradigms*, NSPW '04, pages 67–77, New York, NY, USA, 2004. ACM.

[50] A.-R. Sadeghi, I. Visconti, and C. Wachsmann. Enhancing rfid security and privacy by physically unclonable functions. In *Towards Hardware-Intrinsic Security*, pages 281–305. Springer, 2010.

[51] S. Schüppen, D. Teubert, P. Herrmann, U. Meyer, and S. Sch. Fanci: feature-based automated nxdomain classification and intelligence. In *Proceedings of the 27th USENIX Conference on Security Symposium*, pages 1165–1181. USENIX Association, 2018.

[52] A. Seshadri, M. Luk, A. Perrig, L. V. Doorn, and P. Khosla. SCUBA: Secure Code Update By Attestation in Sensor Networks. In *ACM WiSec*, 2006.

[53] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla. Using FIRE & ICE for Detecting and Recovering Compromised Nodes in Sensor Networks. Technical report, DTIC Document, 2004.

[54] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. Van Doorn, and P. Khosla. Pioneer: Verifying Code Integrity and Enforcing Untampered Code Execution on Legacy Systems. In *ACM SIGOPS OSR*, 2005.

[55] U. Shankar, M. Chew, and J. D. Tygar. Side effects are not sufficient to authenticate software. In *USENIX Security*, May 2004.

[56] A. K. Sikder, H. Aksu, and A. S. Uluagac. 6thsense: A context-aware sensor-based attack detector for smart devices. In *USENIX Security*, 2017.

[57] R. P. Stanley. *Enumerative Combinatorics*, volume 1. Cambridge University Press, second edition, 2012. pp. 64-67.

[58] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ICS) security. *NIST special publication*, 800(82):16–16, 2011.

[59] R. Strackx, F. Piessens, and B. Preneel. Efficient Isolation of Trusted Subsystems in Embedded Systems. In *SecureComm*. Springer, 2010.

[60] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1092–1105. ACM, 2016.

[61] J. Valente, C. Barreto, and A. A. Cárdenas. Cyber-physical systems attestation. In *2014 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 354–357. IEEE, 2014.

[62] O. Willers, C. Huth, J. Guajardo, and H. Seidel. Mems gyroscopes as physical unclonable functions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 591–602. ACM, 2016.

[63] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

10 Appendices

10.1 The Aging Effect on Classification Performance

The aging effect was reported in many PUF applications at the semiconductor level [38, 46]. However, such an effect is not common in mechanical actuators (the preciseness guarantee of the actuators could be found in the actuators data-sheet). We evaluated the aging effect by considering two data-sets of 6 months ago (old data-set) and a recent data-set (recent data-set). As we could see in Table 4 the tuned classifier would provide better classification results. We would recommend performing the tuning of the classifier by including the recent normal traces during the idle time of the CPS. However, the performance of the classifier that is built by training the old data-set is sufficient for the robot arm use-case that we evaluated in this paper. Figure 6 shows the true positive rate

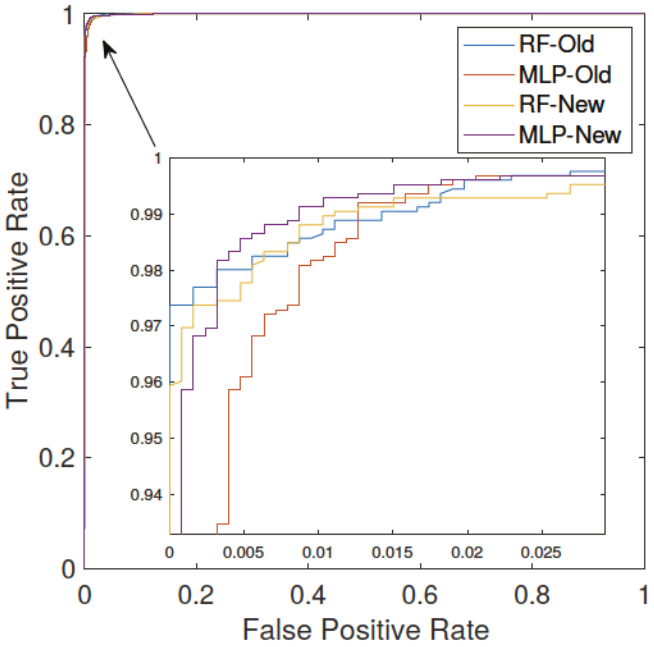


Figure 6: ROC curve of true positive rate (sensitivity) against false positive rate (1-precision), considering the aging effect.

(sensitivity) against the false positive rate (1-precision) in aging effect evaluation. As we could see in Figure 6, the classification performance of the classifiers with the two data-set of old and recent sensor traces are close to each other.

10.2 Performance Metrics

To evaluate the performance of the proposed method, we used eight performance metrics. The true positive (TP) is the number of retrieved relevant instances. The false positive (FP) is the number of retrieved nonrelevant instances. The true negative (TN) is the number of not retrieved nonrelevant instances. The false negative (FN) is the number of not retrieved relevant instances. The Sensitivity rate (Recall, eq. 6) presents the rate of retrieved relevant instances (TP) in overall relevant instances (TP + FN). The Precision rate (specificity, eq. 7) demonstrate the fraction of relevant instances (TP) in overall retrieved instances (TP + FP).

$$\text{Sensitivity rate} = \frac{TP}{TP+FN} \tag{6}$$

$$\text{Precision rate} = \frac{TP}{TP+FP} \tag{7}$$

The false positive rate (eq. 8) is the rate of retrieved nonrelevant instances (FP) in overall nonrelevant instances (FP + TN). The false negative rate (eq. 9) is the rate of not retrieved relevant instances (FN) in overall relevant instances (FN + TP). The false discovery rate (eq. 10) is the rate of retrieved nonrelevant instances (FP) in overall retrieved instances (FP + TP).

$$\text{False positive rate} = \frac{FP}{FP+TN} \tag{8}$$

$$\text{False negative rate} = \frac{FN}{FN+TP} \tag{9}$$

$$\text{False discovery rate} = \frac{FP}{FP+TP} \tag{10}$$

The accuracy (eq. 11) is the rate of retrieved relevant instances and not retrieved nonrelevant instances (TP + TN) in overall instances (TP + TN + FP + FN).

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{11}$$

The F1-score (eq. 12) is a metric for the test's accuracy. The F1-score (also F-score or F-measure) is defined as follows:

$$F1-\text{score} = \frac{2 \times \text{Sensivity} \times \text{Precision}}{\text{Sensivity} + \text{Precision}} \tag{12}$$

The Matthews correlation coefficient (MCC) is a metric for the quality of two-class classification. The MCC metric is one of the most interesting metrics in anomaly detection where the physical feature will be classified to normal and abnormal classes. The MCC is defined as follows:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}$$

10.3 Decoding ROC

Our ROC (true positive rate (sensitivity) against the false positive rate (1-precision)) for the decoding classifiers is presented in Figure 7.

10.4 An Example of Path Strategy

Figure 8 represents the path strategy of `10100010`, and the path strategy of `00000111`. We know that the number of unique paths u in a $x \times y$ grid can be computed as follows [57]:

$$u = \frac{(x+y)!}{(x!y!)} \tag{13}$$

Thus, we can enumerate all possible paths, and use an integer between 1 and u as an index to represent a specific path strategy in a $x \times y$ grid.

Table 4: Performance comparison of different classifiers with different metrics of Sensitivity & Specificity & Precision & FPR & FNR & Accuracy & F1-score & MCC (Matthews Correlation Coefficient), classifiers: Random Forest (RF) & Multilayer Perceptron (MLP).

Algorithms	Sensitivity	Specificity	Precision	FPR	FNR	Accuracy	F1-score	MCC
RF (old)	0.9857	0.9913	0.9912	0.0087	0.0143	0.9885	0.9884	0.9770
MLP (old)	0.9952	0.9826	0.9827	0.0174	0.0048	0.9889	0.9889	0.9779
RF (recent)	0.9912	0.9897	0.9897	0.0103	0.0088	0.9905	0.9905	0.9810
MLP (recent)	0.9881	0.9905	0.9904	0.0095	0.0119	0.9893	0.9892	0.9786

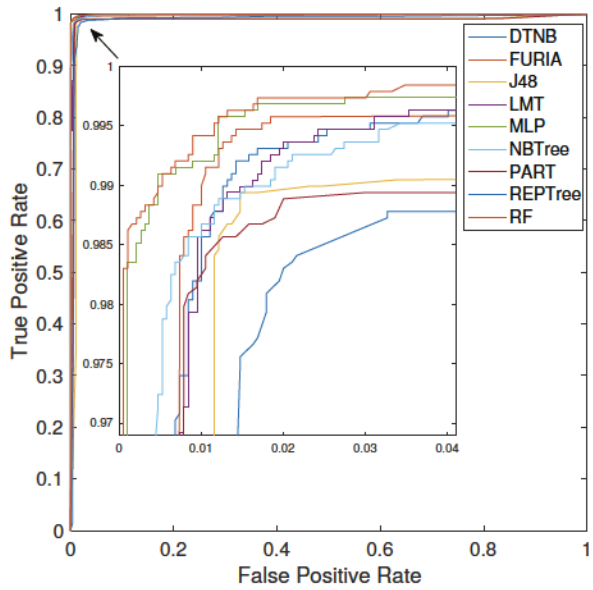


Figure 7: ROC curve of true positive rate (sensitivity) against false positive rate (1-precision).

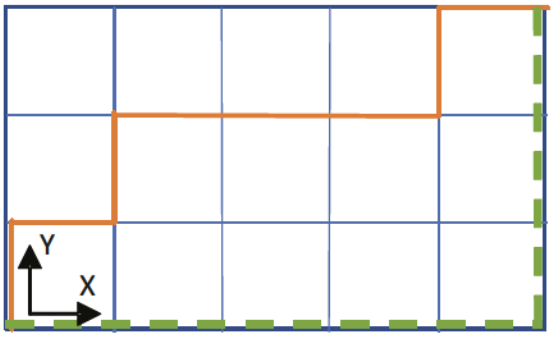


Figure 8: Two examples of path strategies: red=10100010, dashed green=00000111.