POSTER: Towards Cyber Resilience of Cyber-Physical Systems using Tiny Twins

Fereidoun Moradi[*]
Mälardalen University
Västerås, Sweden
fereidoun.moradi@mdu.se

Sara Abbaspour Asadollah
Mälardalen University
Västerås, Sweden
sara.abbaspour@mdu.se

Marjan Sirjani
Mälardalen University
Västerås, Sweden
marjan.sirjani@mdu.se

***Abstract*—We propose a method to detect attacks on sensor data and control commands in cyber-physical systems. We develop a monitor module that uses an abstract digital twin, Tiny Twin, to detect false sensor data and faulty control commands. The Tiny Twin is a state transition system that represents the observable behavior of the system. The monitor observes the sensor data and the control commands transmitted in the network, walks over the Tiny Twin and checks whether the observed data and commands are consistent with the transitions in the Tiny Twin. The monitor produces an alarm when an attack is detected. The Tiny Twin is built automatically based on a timed actor code of the system. We demonstrate the method and evaluate it in detecting attacks using a temperature control system.**

***Index Terms*—Monitoring, Model Checking, Cyber-Physical Systems, Attack Detection and Prevention, Cyber-Security**

1. Introduction

Cyber-Physical Systems (CPSs) are safety-critical systems that integrate physical processes in the industrial plants (e.g., thermal power plants or smart water treatment plants) with sensors, actuators and controller components. Since these components are integrated via a communication network (usually wireless), a CPS is vulnerable to malicious cyber-attacks that may cause catastrophic damage to the physical infrastructure and processes. Cyber-attacks may be performed over a significant number of attack points and in a coordinated way. So, detecting and preventing attacks in CPSs are of significant importance.

Intrusion Detection Systems (IDSs) are deployed in communication networks to defend the system against cyber-attacks. Regular IDSs cannot easily catch complex attacks. They need to be equipped with complicated logic, based on human (safety and security engineers) reasoning [1]. In rule-based IDSs [1], a set of properties that are extracted from the system design specification are considered as rule-sets to detect attacks. Indeed, if an IDS finds a deviation between the observed packets in the network and the defined rules, it produces an alarm and takes a predefined action such as dropping the packets. The key challenge is the effort required to specify the correct system behavior as rules.

2. Overview

We propose a method to detect cyber-attacks on sensor data and control commands in CPSs. The overview of

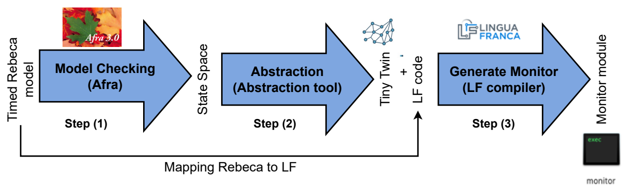


Figure 1: The overview of our method. Step (1), we generate the state space of the Timed Rebeca code of a CPS by the Afra model checker (see Sec. 3). Step (2), the state space is abstracted by our abstraction tool (see Sec. 4). The result of the abstraction is a Tiny Twin that is used within a monitor module (see Sec. 5) to detect the attacks. Step (3), we develop the monitor module in LF language and use the LF compiler to generate an executable file.

our method is shown in Figure 1. The model of a CPS is developed in Timed Rebeca [2] and the Afra model checker [3] is used to verify the model (step (1) in Figure 1). In [4], it is shown that how entities of a CPS, i.e., sensors, actuators, controllers, and physical plant are modeled as actors, and interactions between them are modeled as messages passed between the actors. We develop an abstraction tool (step (2) in Figure 1) that abstracts the state space of the Timed Rebeca model (generated by Afra model checking tool) to create a Tiny Twin (TT) [5]. Digital Twins (DT) [6] are used as digital representation of the system to advance the system monitoring. We develop a monitor module that uses the created Tiny Twin to track the order and the timing of events (step (3) in Figure 1).

3. Model Checking and Security Analysis

We assess the security aspects of a CPS by verifying its security properties. Afra supports LTL, TCTL and assertions for property specification. By model checking we analyze security of the CPS design to recognize where the potential attack scenarios can successfully cause a failure in the system. We utilize the STRIDE [7] model as a reference for classifying potential attacks on a CPS. In Table 1, we classify the significant attacks on CPS based on the STRIDE categories. The cyber and physical attacks exploit emerging CPS-related vulnerabilities in the two aspects of *communication* and *component*, and are shown in Table 1 as *Scheme-A* and *Scheme-B*. *Scheme-A* consists of the attack scenarios which are secretly recording or modifying the data transmitted over the channels (e.g., eavesdropping, MITM and injection attack). *Scheme-B* includes the attacks that inject malicious code into the software components or perform a malicious alteration on a physical component (e.g., malware and physical attack).

TABLE 1: Attack Classification according STRIDE threat modeling [4].

Threat Type	*Cyber or Physical Attack*	*Scheme-A*	*Scheme-B*
Spoofing (Authentication)	Masquerade attack		[8]
	Packet spoofing attack	[9]	
Tampering (Integrity)	Man-in-the-middle (MITM)	[8]	
	Injection attack	[9]	[9]
	Replay attack	[8]	
	Malware (Virus or Worms)		[9]
	Physical attack	[9]	[10]
Reputation (Non-Repudiation)	On-Off attack		[10]
Information Disclosure (Confidentiality)	Eavesdropping	[8]	
	Malware (Spyware)		[9]
	Side-channel attack		[9]
	Physical attack	[9]	[10]
Denial of Service (Availability)	Resource exhaustion attack	[8]	[9]
	Interruption attack	[8]	
	Malware (Ransomware)		[9]
	Physical attack	[9]	[10]
Elevation of Privilege (Authorization)	Malware (Rootkit)		[9]

4. Abstraction and Tiny Twin

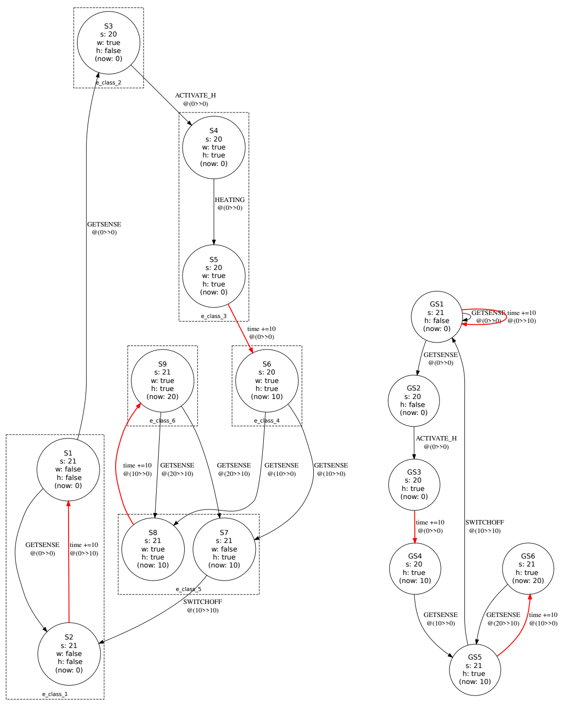
To create a Tiny Twin, we abstract the given state space with respect to the sensor data and control commands. We abstract the state space generated by the model checker in order to preserve sequences of observable actions while hiding internal actions. Our abstraction method is implemented in a tool by considering the reduction algorithm in [11]. It is applied to the original state space where it iteratively refines indistinguishable states, i.e., the classes containing equivalent states, while hides transitions that are called silent transitions. Figure 2 illustrates how the abstraction tool performs on an example.

The Tiny Twin defines the observable behavior of the system in the absence of an attack and contains the order and the time at which the sensor data and control commands are communicated. Transitions in Tiny Twin are tagged by a label that indicates an action or the progress of time.

5. Monitoring and Attack Detection

We develop a monitor module that observes the sensor data and the control commands transmitted in the network and decides to drop or pass the control commands to the actuators (see Figure 3). The Tiny Twin is placed within the module and serves as a baseline for detecting attacks. The module starts its observation when the system executes. Upon receiving sensor data, the module compares it with the state variables in the Tiny Twin. If the module finds no differences between them, it proceeds and checks whether the commands are consistent with the corresponding transitions in the model. If this is the case, the module sends the commands to the actuators. Otherwise, the module produces an alarm and terminates the process of monitoring.

We implement our monitor module in Lingua Franca (LF) [12] that is a language for programming CPSs. In principle, a Lingua Franca code can connect to the physical plant and the controller through the input/output communication channels in the actual system. We use Lingua Franca to simulate the system at runtime and evaluate the detection capability of our method by defining compromised components.


S3 s: 20 w: true h: false (now: 0)
e_class_2
ACTIVATE_H @(0>>0)
S4 s: 20 w: true h: true (now: 0)
HEATING @(0>>0)
S5 s: 20 w: true h: true (now: 0)
e_class_3
GETSENSE @(0>>0)
time +=10 @(0>>0)
S9 s: 21 w: true h: true (now: 20)
e_class_6
S6 s: 20 w: true h: true (now: 10)
e_class_4
S1 s: 21 w: false h: false (now: 0)
S8 s: 21 w: true h: true (now: 10)
S7 s: 21 w: false h: true (now: 10)
e_class_5
SWITCHOFF @(10>>10)
S2 s: 21 w: false h: false (now: 0)
e_class_1
GS1 s: 21 h: false (now: 0)
GS2 s: 20 h: false (now: 0)
ACTIVATE_H @(0>>0)
GS3 s: 20 h: true (now: 0)
GS4 s: 20 h: true (now: 10)
GS5 s: 21 h: true (now: 10)
GS6 s: 21 h: true (now: 20)
SWITCHOFF @(10>>10)
(a)
(b)
Figure 2: (a) The transition system of an example Timed Rebeca model with the equivalence classes created by the abstraction tool. (b) The Tiny Twin of the transition system depicted in Figure 2(a) [5].

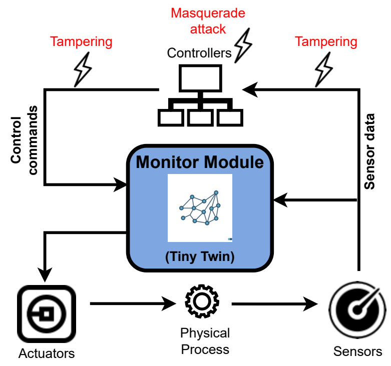


Figure 3: The monitor module observes the input/output of the controllers and drops faulty control commands if it identifies a mismatch between the state transitions in Tiny Twin and the observed sensor data and control commands.

6. Case Study: a Temperature Control System

We evaluate our method in detecting and preventing cyber-attacks using a temperature control system. The temperature control system is responsible for maintaining the temperature of a room at a desired range (e.g., the values between 21 and 23). This system includes a sensor, a hc_unit (heating and cooling unit) as an actuator, and a controller. The controller receives sensor data from the sensor and transmits the activate_c, activate_h or switch off command to the hc_unit to respectively activate the cooling or heating process, or switch the heating/cooling process off. We use the Afra model checker to produce the state space of the developed Timed Rebeca model and exploit our abstraction tool to generate the Tiny Twin. We implement both the system and the monitor module in LF.

6.1. Tiny Twin

We create the Tiny Twin of the state space of the developed Timed Rebeca model for the temperature control system. The Tiny Twin is generated by the abstraction tool based on the list V={*sensedValue*, *cooler_on*, *heater_on*} of state variables. The original state space of the model includes 76 states and 103 transitions while the generated Tiny Twin contains 21 states (i.e., equivalence classes) and 36 transitions. The Tiny Twin is trace equivalent to the original state space (projected on the variables containing sensors data and control commands).

6.2. Attack Types and Detection Capability

We evaluate the capability of the developed monitor module in detecting the attacks. We consider three types of attacks that target the *integrity* aspect of CPS (see Figure 3). (1) Attackers have the ability of tampering sensor data or injecting any arbitrary values into the vulnerable channel between controller and sensors, i.e., *replay or tampering attack*, (2) attackers are able to manipulate the controller through malicious code injection into the software of the controller, i.e., *fabrication or masquerade attack*, and (3) one or more attackers can perform a coordinated attack to force the system to change its correct functionally.

We consider the number of false sensor data and faulty control commands sent by the compromised components as the number of attacks. In our experiments, we simulate 20 false sensor data and 12 faulty control commands as listed in Table 2. We also simulate 240 coordinated attacks (combination of the false sensor data and the faulty control commands). We calculate the detection rate of the monitor with respect to the detected/undetected attacks. In this case study, the detection rate is around 68.8 percent.

TABLE 2: Attacks and detection capability of the monitor [5].

System States	*# Attacks*	*False sensor data/ Faulty control commands*	*Detection Capability* (DS/DC)
GS1 and GS2	4	Sensor data (20, 21, 23, or 24)	DS (20 and 24)
GS3 and GS5	4	Sensor data (20, 21, 22, or 24)	DS (20 and 21)
GS4 and GS6	4	Sensor data (20, 22, 23, or 24)	DS (23 and 24)
GS8	2	Command (*activate_h* or *switchoff*)	DC (*activate_h* and *switchoff*)
GS9	2	Command (*activate_c* or *switchoff*)	DC (*activate_c* and *switchoff*)
GS11 and GS13	4	Sensor data (20, 21, 22, or 23)	DS (20 and 21)
GS12 and GS14	4	Sensor data (21, 22, 23, or 24)	DS (23 and 24)
GS15, GS16, GS17, GS18	2	Command (*activate_h* or *activate_c*)	DC (*activate_h* and *activate_c*)

#Attacks.: Number of simulated attacks, *DS*: Detect false sensor data, *DC*: Detect faulty control commands.

Table 3 shows the alarms list returned by the monitor module when a false sensor data or a faulty control command is detected. The alarm is comprised of a time value, a false sensor data or a faulty control command, the status of the physical plant reported by the sensor and the value of the state variables in the state where the monitor module terminated the system execution.

TABLE 3: Alarms of the monitor module in case of attacks [5].

System States	*False sensor data/ Faulty control commands*	*Alarms list*
GS1 and GS2	Sensor data (20)	$[time, y^i : 20, y : 23, s : 22, c : false, h : false]$
GS3 and GS5	Sensor data (21)	$[time, y^i : 21, y : 22, s : 23, c : false, h : false]$
GS4 and GS6	Sensor data (23)	$[time, y^i : 23, y : 22, s : 23, c : false, h : false]$
GS8	Command (*activate_h*)	$[time, u^d : activate\_h, y : 24, s : 24, c : false, h : false]$
GS9	Command (*switchoff*)	$[time, u^d : switchoff, y : 20, s : 20, c : false, h : false]$
GS11 and GS13	Sensor data (21)	$[time, y^i : 21, y : 22, s : 24, c : true, h : false]$
GS12 and GS14	Sensor data (24)	$[time, y^i : 24, y : 22, s : 20, c : false, h : true]$
GS16	Command (*activate_c*)	$[time, u^d : activate\_c, y : 22, s : 22, c : true, h : false]$

time: The logical time which is derived using Lingua Franca code.

In a CPS, there may be several variables involved in the physical process as well as various sensors and actuators. The monitoring approach using the Tiny Twin enables us to consider only variables are affected during an attack (i.e., violation of properties). Tiny Twin provides relevant information about attacks that can be employed in mitigation techniques, backtracking and recovering the system after attacks. We have developed the models and the LF codes of two case studies (Pneumatic Control System and Secure Water Treatment system), for which the *monitor* module can properly detect the attacks on the system.

7. Conclusion and Future Work

In this work, we proposed a method for detecting cyber-attacks on CPSs. In particular, we used a Tiny Twin to detect the attacks on sensor data and control commands. We developed an abstraction tool to build the Tiny Twin, which is an abstract version of a state transition system representing the system correct behavior in the absence of an attack. In our method, we built a monitor module that executes together with the system. It produces an alarm if the sensor data or the control commands are not consistent with the state transitions in the Tiny Twin. We evaluated the capability of our method in detecting attacks on a temperature control system. As the future work, we aim to build a module to mitigate impacts of the attacks based on the predefined mitigation plans.

References

[1] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[2] M. Sirjani and E. Khamespanah, "On time actors," in *Theory and Practice of Formal Methods*, pp. 373–392, Springer, 2016.

[3] "Afra: an integrated environment for modeling and verifying rebeca family designs." https://rebeca-lang.org/alltools/Afra, 2022. [Online; accessed Feb 09, 2022].

[4] F. Moradi, S. A. Asadollah, A. Sedaghatbaf, A. Čaušević, M. Sirjani, and C. Talcott, "An actor-based approach for security analysis of cyber-physical systems," in *International Conference on Formal Methods for Industrial Critical Systems*, pp. 130–147, Springer, 2020.

[5] F. Moradi, M. Bagheri, H. Rahmati, H. Yazdi, S. A. Asadollah, and M. Sirjani, "Monitoring cyber-physical systems using a tiny twin to prevent cyber-attacks," in *International Symposium on Model Checking of Software (SPIN)*, 2022.

[6] M. Eckhart and A. Ekelhart, "A specification-based state replication approach for digital twins," in *Proceedings of the 2018 workshop on cyber-physical systems security and privacy*, pp. 36–47, 2018.

[7] A. Shostack, *Threat modeling: Designing for security*. Wiley, 2014.

[8] S. Choi, J.-H. Yun, and S.-K. Kim, "A comparison of ics datasets for security research based on attack paths," in *International Conference on Critical Information Infrastructures Security*, Springer, 2018.

[9] J.-M. Flaus, *Cybersecurity of industrial systems*. J. Wiley & Sons, 2019.

[10] 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 Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[11] D. N. Jansen, J. F. Groote, J. J. Keiren, and A. Wijs, "A simpler o (m log n) algorithm for branching bisimilarity on labelled transition systems," *arXiv preprint arXiv:1909.10824*, 2019.

[12] M. Lohstroh, C. Menard, A. Schulz-Rosengarten, M. Weber, J. Castrillon, and E. A. Lee, "A language for deterministic coordination across multiple timelines," in *2020 Forum for Specification and Design Languages (FDL)*, pp. 1–8, IEEE, 2020.

Towards Cyber Resilience of Cyber-Physical Systems using Tiny Twins

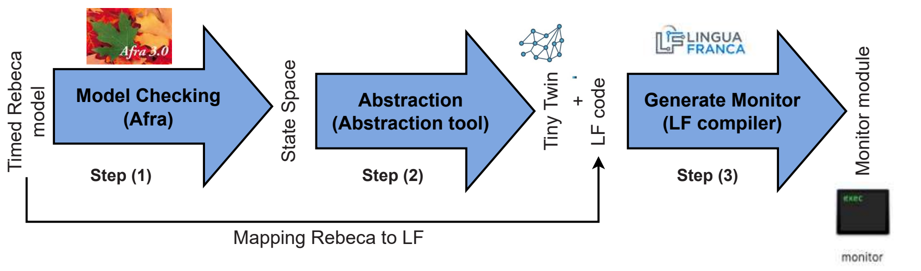
Fereidoun Moradi*, Marjan Sirjani, Sara Abbaspour and Maghsood Salimi

School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden

fereidoun.moradi@mdu.se

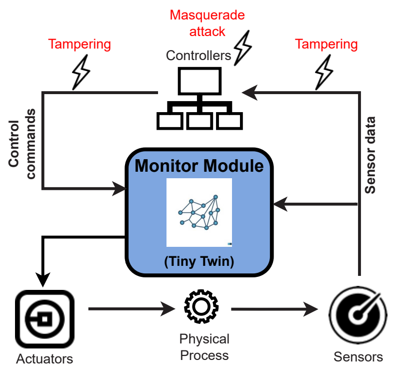
Overview of our Method

We model system in Timed Rebeca and use Afra model checker to verify the model (step 1). We develop an abstraction tool that reduces the state space of the model and creates a Tiny Twin (step 2). We develop a monitor module that observes sensor data and control commands and uses the created Tiny Twin to track the order and the timing of the observable actions (step 3).



Monitoring

The monitor observes the input/output of the controllers and detects faulty control commands.



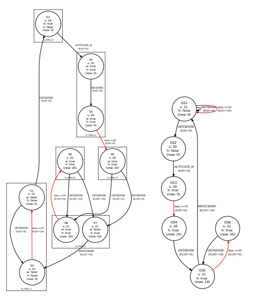
Security Analysis

The STRIDE threat modeling is used as a reference for classifying potential attacks on cyber-physical systems.

Threat Type	*Cyber or Physical Attack*	*Comm.*	*Comp.*
Spoofing (Authentication)	Masquerade attack		✓
	Packet spoofing attack	✓	
Tampering (Integrity)	Man-in-the-middle (MITM)	✓	
	Injection attack	✓	✓
	Replay attack	✓	
	Malware (Virus or Worms)		✓
	Physical attack	✓	✓
Reputation (Non-Repudiation)	On-Off attack		✓
Information Disclosure (Confidentiality)	Eavesdropping	✓	
	Malware (Spyware)		✓
	Side-channel attack		✓
	Physical attack	✓	✓
Denial of Service (Availability)	Resource exhaustion attack	✓	✓
	Interruption attack	✓	
	Malware (Ransomware)		✓
	Physical attack	✓	✓
Elevation of Privilege (Authorization)	Malware (Rootkit)		✓

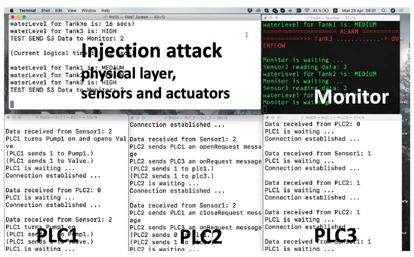
Create a Tiny Twin

The Tiny Twin is an abstract model which defines the observable behavior of the system in the absence of an attack and preserves branching bisimulation relation with the original model.



Case-Studies

We develop the Timed Rebeca models and the Lingua Franca (LF) codes of two case studies, a water treatment system and a pneumatic control system, for which the monitor module can properly detect coordinated and complex attacks on the system.



```
waterLevel for Tank(me is: 16 secs)
waterLevel for Tank3 is: HIGH
TEST SEND S3 Data to Monitor: 2

(Current logical time is: ... secs)

waterLevel for Tank1 is: MEDIUM
waterLevel for Tank2 is: MEDIUM
waterLevel for Tank3 is: HIGH
TEST SEND S3 Data to Monitor: 2
```

```
waterLevel for Tank1 is: MEDIUM
=================== ALARM =========
==========>>> Tank3 ..........-> OV
ERFLOW

Monitor is waiting ...
Sensor2 reading data: 2
waterLevel for Tank2 is: MEDIUM
Monitor is waiting ...
Sensor3 reading data: 2
waterLevel for ...
Monitor is waiting ...
```

```
Data received from Sensor1: 2
PLC1 turns Pump1 on and opens Val
ve
(PLC1 sends 1 to Pump1.)
(PLC1 sends 1 to Valve.)
PLC1 is waiting ...
Connection established ...

Data received from PLC2: 0
PLC1 is waiting ...
Connection established ...

Data received from Sensor1: 2
PLC1 turns Pump1 on
(PLC1 sends 1 to Pump1.)
(PLC1 sends 1 to Valve.)
PLC1 is waiting ...
```

```
Connection established ...

Data received from Sensor1: 2
PLC2 sends PLC1 an openRequest messa
ge
PLC2 sends PLC3 an onRequest message
(PLC2 sends 1 to plc1.)
(PLC2 sends 1 to plc3.)
PLC2 is waiting ...
Connection established ...

Data received from Sensor1: 2
PLC2 sends PLC1 an closeRequest mess
age
PLC2 sends PLC3 an onRequest message
(PLC2 sends 0 to plc1.)
(PLC2 sends 1 to plc3.)
PLC2 is waiting ...
```

```
Data received from PLC2: 0
PLC1 is waiting ...
Connection established ...

Data received from Sensor1: 1
PLC1 is waiting ...
Connection established ...

Data received from PLC2: 1
PLC1 is waiting ...
Connection established ...

Data received from PLC2: 1
PLC1 is waiting ...
Connection established ...

Data received from Sensor1: 1
PLC1 is waiting ...
```

Impact and Contribution

Defining policies for detecting attacks is a manual and challenging task for safety/security engineers. We make the attack detection process systematic and automatic, and provide tractable reports after a successful detection. We employ a Tiny Twin to detect cyber-attacks including complex attacks.