Mälardalen University Doctoral Dissertation 407

Automated Vulnerability Discovery and Attack Detection Framework for Cyber-Physical Systems

Fereidoun Moradi

Mälardalen University

Mälardalen University Press Dissertations
No. 407

AUTOMATED VULNERABILITY DISCOVERY AND ATTACK DETECTION FRAMEWORK FOR CYBER-PHYSICAL SYSTEMS

Fereidoun Moradi

2024

School of Innovation, Design and Engineering

Copyright © Fereidoun Moradi, 2024
ISBN 978-91-7485-646-0
ISSN 1651-4238
Printed by E-Print AB, Stockholm, Sweden

Mälardalen University Press Dissertations
No. 407

AUTOMATED VULNERABILITY DISCOVERY AND ATTACK DETECTION FRAMEWORK FOR CYBER-PHYSICAL SYSTEMS

Fereidoun Moradi

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid
Akademin för innovation, design och teknik kommer att offentligen försvaras
fredagen den 7 juni 2024, 13.15 i Omega, Mälardalens universitet, Västerås.

Fakultetsopponent: Professor Massimo Merro, University of Verona, Italy

Akademin för innovation, design och teknik

Abstract

The combination of physical assets with cyber computational entities, known as Cyber-Physical Systems (CPS), is becoming more common. These systems are used in various sectors such as manufacturing, energy production, and transportation. However, they may contain vulnerabilities that can be exploited and compromise the proper functioning of the systems. Formal methods can provide confidence in the correctness of the system. These methods can help system designers to discover vulnerabilities and ensure the system meets its intended behavior and properties.

This thesis presents a framework called CRYSTAL for analyzing the security of CPS both at the design and runtime with the following contributions: 1) We build a Timed Rebeca model augmented with attack scenarios to identify potential vulnerabilities during the design phase. We identify CPS-related attacks reported in the literature and use STRIDE threat modeling as a guideline to put them into two categories: attacks on communication and attacks on components. We augment our model with actors representing attackers that jeopardize the communication and compromised components with possible malfunctions. Subsequently, we analyze the security of the CPS design (i.e., a model augmented with attacks) using model checking and identifying the trace of events leading to a security failure (i.e., counterexamples). 2) We propose a model-based methodology for detecting attacks at runtime. We develop a monitor and an algorithm that checks whether the behavior of the system at runtime is consistent with a reference model. We call the reference model as Tiny Digital Twin and it is employed within the monitor. We create Tiny Digital Twin by automatically reducing the state space of the Timed Rebeca model while preserving the trace equivalence. 3) We provide a formal foundation for our abstraction method and present a theory to map the state space of a Timed Rebeca model into a Labeled Transition System (LTS). This is achieved by defining a Concise Rebeca Timed Transition System (CRTTS) and implementing a function to convert CRTTS into an LTS. We use mCRL2 ltsconvert tool to abstract away non-observable actions from LTS while preserving trace equivalence between the original model and its abstracted version. 4) To validate the effectiveness of our attack detection method we simulate various attacks. Subsequently, we systematically model and check complex coordinated attacks. The applicability of CRYSTAL is demonstrated in revealing malicious behavior within different case studies including Pneumatic Control System (PCS), Temperature Control System (TCS), and Secure Water Treatment System (SWaT).

ISBN 978-91-7485-646-0
ISSN 1651-4238

Abstract

The combination of physical assets with cyber computational entities, known as Cyber-Physical Systems (CPS), is becoming more common. These systems are used in various sectors such as manufacturing, water supply systems, energy production, and transportation. However, they may contain vulnerabilities that can be exploited and compromise the proper functioning of the systems. The vulnerabilities can result from the lack of security measures in the design of the system as well as the use of insecure communication channels between the cyber and physical components. Formal methods can provide confidence in the correctness of the system. These methods can help system designers to discover vulnerabilities and ensure the system meets its intended behavior and properties.

This thesis presents a framework called CRYSTAL for analyzing the security of CPS both at the design and runtime with the following contributions: 1) We build a Timed Rebeca model augmented with attack scenarios to identify potential vulnerabilities during the design phase. We identify CPS-related attacks reported in the literature and use STRIDE threat modeling as a guideline to put them into two categories: *attacks on communication* and *attacks on components*. We augment our model with actors representing attackers that jeopardize the communication and compromised components with possible malfunctions. Subsequently, we analyze the security of the CPS design (i.e., a model augmented with attacks) using model checking and identifying the trace of events leading to a security failure (i.e., counter-examples). 2) We propose a model-based methodology for detecting attacks at runtime. We develop a monitor and an algorithm that checks whether the behavior of the system at runtime is consistent with a reference model. We call the reference model as *Tiny Digital Twin* and it is employed within the monitor. We create Tiny Digital Twin by automatically reducing the state space of the Timed Rebeca model while preserving the trace equivalence.

3) We provide a *formal foundation* for our abstraction and present a theory to map the state space of a Timed Rebeca model into a Labeled Transition System (LTS). This is achieved by defining a Concise Rebeca Timed Transition System (CRTTS) and implementing a function to convert CRTTS into an LTS. We use mCRL2 *ltsconvert* tool to abstract away non-observable actions from LTS while preserving trace equivalence between the original model and its abstracted version. 4) To validate the effectiveness of our attack detection method we simulate various attacks. We systematically model and check *complex coordinated attacks*. The applicability of CRYSTAL is demonstrated in revealing malicious behavior within different case studies including Pneumatic Control System (PCS), Temperature Control System (TCS), and Secure Water Treatment System (SWaT).

Sammanfattning

Kombinationen av fysiska tillgångar med cyberberäkningsenheter, kända som Cyber-Physical Systems (CPS), blir allt vanligare. Dessa system används inom olika sektorer såsom tillverkning, vattenförsörjningssystem, energiproduktion och transport. Dock kan de innehålla sårbarheter som kan utnyttjas och komprometttera systemens korrekta funktion.Sårbarheterna kan bero på bristande säkerhetsåtgärder i systemets design samt användningen av osäkra kommunikationskanaler mellan cyber- och fysiska komponenter. Formella metoder kan ge förtroende för systemets korrekthet. Dessa metoder kan hjälpa systemdesigners att upptäcka sårbarheter och säkerställa att systemet uppfyller sitt avsedda beteende och egenskaper.Denna avhandling presenterar en ram kallad CRYSTAL för att analysera säkerheten hos CPS både i design- och driftfaserna. Våra bidrag är följande: a) Vi bygger en Timed Rebeca-modell förstärkt med attackscenarier för att identifiera potentiella sårbarheter under designfasen. Vi identifierar först CPS-relaterade attacker som rapporterats i litteraturen och använder STRIDE-hotmodellering som riktlinje för att kategorisera dem i två kategorier: attacker på kommunikation och attacker på komponenter. Sedan använder vi Timed Rebeca-modellspråket för att modellera en aktör som en angripare som äventyrar kommunikationen och en aktör som en komprometterad komponent med möjliga fel. Därefter analyserar vi säkerheten hos CPS-designen (en modell förstärkt med attacker) genom modellkontroll och identifierar spåret av händelser som leder till ett säkerhetsfel (kontra-exempel). b) Vi föreslår en modellbaserad metodik för att upptäcka attacker vid drifttid. Inledningsvis minskar vi automatiskt tillståndsutrymmet för en Timed Rebeca-modell och skapar en abstrakt modell kallad Tiny Digital Twin. Vi bevarar spår-ekvivalensen mellan det ursprungliga tillståndsutrymmet och dess abstrakta version. Därefter utvecklar vi en övervakningsalgoritm implementerad i en modul som använder Tiny Digital Twin för att kontrollera om systemets beteende vid drifttid är konsistent med modellen. Övervakaren producerar ett larm när en inkonsekvens upptäcks. c) Vi tillhandahåller en formell grund för vår metodik och presenterar en teori för att kartlägga tillståndsutrymmet för en Timed Rebeca-modell till ett Labeled Transition System (LTS). Detta uppnås genom att definiera ett Concise Rebeca Timed Transition System (CRTTS) och implementera en ltscast-funktion för att konvertera CRTTS till ett LTS. Vi använder mCRL2 ltsconvert-verktyget för att abstrahera bort icke-observerbara handlingar från LTS samtidigt som vi bevarar spår-ekvivalensen mellan den ursprungliga modellen och dess abstrakta version. d) För att validera effektiviteten hos vår metod för att upptäcka attacker simulerar vi olika attacker. Vi modellerar systematiskt och kontrollerar komplexa koordinerade attacker. CRYSTAL:s tillämplighet demonstreras genom att avslöja skadligt beteende inom olika fallstudier, inklusive Pneumatisk Kontrollsystem (PCS), Temperaturkontrollsystem (TCS) och Säkert Vattenbehandlingssystem (SWaT).

To my dear family

Anis, Aysu, and Ayden

Acknowledgments

The Ph.D. journey involves a lot of different experiences, some happy and some tough. First, I want to thank my main supervisor, Marjan Sirjani. She has been a big help, encouraging me and putting in a lot of effort to help me improve as a good Ph.D. student. Thanks for being patient, guiding me, and supporting me during my research. You really helped me grow during my Ph.D. I also want to thank Sara Abbaspour, my co-supervisor. She always patiently gave me her thoughts and comments. Thanks, Sara, for your valuable advice, suggestions, and encouragement. Big thanks to Bahman Pourvatan too. He helped me a ton with the last two papers. He was always there to talk things through, give me ideas, and answer my questions. Thanks to Aida Čaušević, one of my earlier co-supervisors. She pushed me to understand the student rules, make a good plan, and choose the right courses for my Ph.D. I also want to thank Radu Dobrin, the head of our division, for making all the administrative stuff easier. Thanks for letting me join the STINT project to visit UTP in Panama and UdeA in Colombia. Thanks to my co-authors, Ali Sedaghatbaf, Hanieh Rahmati, and Zahra Moezkarimi, for working with me on the papers included in this thesis. And of course, thanks to all my colleagues at MDU. Special shoutout to Maghsood Salimi for being a great friend and for all the awesome memories from our trips. Extra thanks to Cristina Seceleanu for reviewing my thesis proposal and giving me helpful feedback. Over the past five years, there were tough times, but thankfully, my family, especially my wife, has been there for me. Without her support, I would not have had the courage to move to Sweden or pursue a career I love. Thank you, Anis, for being my best friend and partner, someone I want to share every moment with. Without you, I could not passed this journey. The work in this thesis was supported by SSF Serendipity project, KKS DPAC Project, and KKS SACSys Synergy project.

Fereidoun Moradi, March 2024, Västerås, Sweden

List of Publications

Papers Included in the Ph.D. Thesis

Paper A: *On-off attack on a blockchain-based IoT system*, **Fereidoun Moradi**, Ali Sedaghatbaf, Sara Abbaspour Asadollah, Aida Čaušević, and Marjan Sirjani. The 24th IEEE Emerging Technologies and Factory Automation (ETFA 2019), First Workshop on Secure and Trustable Wirelessly Connected Industrial IoT, Zaragoza, Spain, 10th - 13th September 2019.

Paper B: *An actor-based approach for security analysis of cyber-physical systems*, **Fereidoun Moradi**, Sara Abbaspour Asadollah, Ali Sedaghatbaf, Aida Čaušević, Marjan Sirjani, and Carolyn Talcott. The 25th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2020), Vienna, Austria, 2nd-3th September 2020.

Paper C: *Monitoring cyber-physical systems using a tiny twin to prevent cyber-attacks*, **Fereidoun Moradi**, Maryam Bagheri, Hanieh Rahmati, Hamed Yazdi, Sara Abbaspour Asadollah, and Marjan Sirjani. The 28th International Symposium on Model Checking Software (SPIN 2022), Virtual Event, May 21, 2022.

Paper D: *Tiny Twins for Detecting Cyber-Attacks at Runtime using Concise Rebeca Time Transition System*, **Fereidoun Moradi**, Bahman Pourvatan, Sara Abbaspour Asadollah, and Marjan Sirjani. Journal of Parallel and Distributed Computing (JPDC), Volume 184, Page 104780, Elsevier, 2023.

Paper E: *CRYSTAL Framework: Cybersecurity Assurance for Cyber-Physical Systems*, **Fereidoun Moradi**, Sara Abbaspour Asadollah, Bahman Pourvatan, Zahra Moezkarimi, and Marjan Sirjani. Journal of Logical and Algebraic Methods in Programming (JLAMP), Volume 139, Page 100965, Elsevier, 2024.

Other Publications[1]

Paper 1: *POSTER: Towards Cyber Resilience of Cyber-Physical Systems using Tiny Twins*, **Fereidoun Moradi**, Sara Abbaspour Asadollah and Marjan Sirjani. The 7th IEEE European Symposium on Security and Privacy (EuroSP 2022), August 2022.

Paper 2: *Enhancing CRYSTAL: Preventive Recovery in Brief*, **Fereidoun Moradi**, Zahra Moezkarimi and Marjan Sirjani. The 34th Nordic Workshop on Programming Theory Program (NWPT 2023), November 2023.

[1]These papers are not included in the Ph.D. thesis.

Contents

I

Thesis

Chapter 1

Introduction

The security of sophisticated industrial systems is becoming increasingly important due to the demand for adaptation in response to unexpected security problems. Cybersecurity concerns are remarkable for large companies in a variety of sectors, such as water treatment systems, transport, electrical grid operations, and smart machines [1, 2, 3]. Many systems are no longer closed and involve cyber and physical systems that work together to achieve a common goal. Despite the advantages of combining cyber and physical components, the openness of some parts of the system makes the whole system exposed to several attacks, which may disrupt the desirable behavior of the system. For example, as reported by Kaspersky ICS CERT [4], some parts of the production line of Japanese Optics manufacturer HOYA in Thailand was crashed by a Malware [5]. Once the malware could spread over a hundred of the company's computers, it stole user credentials and distributed a cryptocurrency miner. The incident also affected computers at HOYA's headquarters in Japan that were connected to the network, disrupting the invoice issue process. Although the company was able to stop the cryptocurrency mining operation, the incident had a substantial impact on the performance of the production management process.

As another example, the attack was launched on the US power facility in March 2019 which caused an interruption in the electrical grid operation [6]. According to the report by North American Electric Reliability Corporation (NERC) [7], the firewall in the network was compromised by an anonymous remote entity. The attackers exploited a known vulnerability in the device which caused the firewall reboots itself in a repetitive manner. During each reboot, the firewall was out of service for about 5 minutes. The direct impact of the attack was communication outages between the control center and devices.

Motivation. To tackle CPS attacks, it is required to consider the security of cyber-physical systems beyond the IT systems standard information security [8, 9]. This issue is more highlighted when utilizing cyber-physical systems in different significant sectors like manufacturing or critical infrastructures, creating a need for efficiently handling relevant security issues.

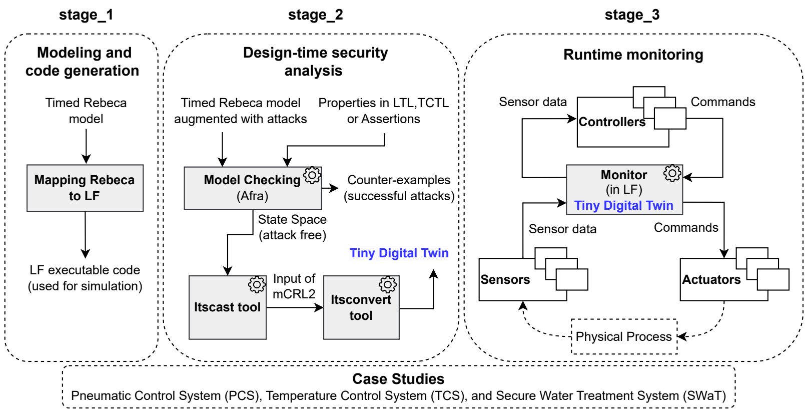
Security assurance is a non-stop process. Companies need to continually assess their cybersecurity posture to ensure they are up to date with the latest security measures. We need to prepare our organizations and industrial companies by using proper tools, solutions and methodologies, both at the design phase and the operational phase of the system, and provide well-formed adaptation strategies to withstand failures. Formal methods provide an approach to verifying software systems, which can be particularly useful in the field of cybersecurity [10]. By using formal methods, one can create a precise mathematical model of the system at design time, which can be used to identify potential vulnerabilities, detect and diagnose flaws and errors, and verify that the system is secure and will behave as intended. Runtime verification and monitoring can also be used for resilience against cyberattacks by preventing and detecting cyberattacks and therefore can help in improving reaction time, reducing downtime, and ultimately saving money in the case of an attack.

CRYSTAL Framework. In this thesis, we present a framework called CRYSTAL to enhance the security of Cyber-Physical Systems (CPS) by detecting disturbances in the system behavior and thus protecting the system against attacks. The development of the CRYSTAL framework is organized as a long-term plan (starting from the research plan at the beginning of the study and ending with the Ph.D. thesis). Therefore, based on the defined goals for the Ph.D. study, we have developed different methods and concepts behind the main idea for the Ph.D. thesis.

The CRYSTAL framework is designed for building safe and secure cyber-physical systems by providing a set of methods and tools for modeling the system, defining attack models, abstracting the model and creating Tiny Digital Twin, and finally developing a monitor to detect cyberattacks. The Tiny Digital Twin is constructed at the design phase and is used for monitoring at the operational phase to ensure that the system is functioning as intended. The architecture of the CRYSTAL is inspired by the MAPE-K [1] feedback loop

[1]The acronym MAPE-K stands for **M**onitor, **A**nalyze, **P**lan, **E**xecute, plus **K**nowledge.

model [11] where we have components to monitor the system, analyze the behavior of the system, plan accordingly, and actuate necessary actions. As shown in Figure 1.1, three stages are defined and developed in the CRYSTAL framework.



stage_1
stage_2
stage_3
Modeling and code generation
Timed Rebeca model
Mapping Rebeca to LF
LF executable code (used for simulation)
Design-time security analysis
Timed Rebeca model augmented with attacks
Properties in LTL,TCTL or Assertions
Model Checking (Afra)
Counter-examples (successful attacks)
State Space (attack free)
Tiny Digital Twin
Itscast tool
Input of mCRL2
Itsconvert tool
Runtime monitoring
Sensor data
Controllers
Commands
Monitor (in LF)
Tiny Digital Twin
Sensor data
Commands
Sensors
Actuators
Physical Process
Case Studies
Pneumatic Control System (PCS), Temperature Control System (TCS), and Secure Water Treatment System (SWaT)

Figure 1.1: Three stages in developing the CRYSTAL framework

Stage_1: Modeling and Code Generation. We use Timed Rebeca as an actor-based modeling language supported by a model checking tool [12, 13, 14] to model the behavior of the cyber-physical systems. We present the components of cyber-physical systems as actors in Timed Rebeca, and define interactions between the components as messages passed between the actors (as shown in Figure 1.2, in a CPS application, controllers, sensors, actuators, and physical processes are modeled as actors).

We map the Timed Rebeca model to a Lingua Franca (LF) executable code [15, 16]. LF is a programming language based on the Reactor model of computation [17] for building CPSs. LF deploys a mechanism to synchronize the logical time (defined in the Timed Rebeca model) with the physical time in the system. The mapping of Timed Rebeca to Lingua Franca and reverse, including the timing features, is a natural mapping that is discussed in [15, 18]. This Lingua Franca code is used to simulate the system behavior when checking the behavior of the system at runtime (throughout the text, the term ’runtime’ refers to the operational phase of the system).

Stage_2: Design-time Security Analysis. In the second stage, we build a Timed Rebeca model that is augmented with attacks to find potential vulnerable points at design time. Figure 1.2 illustrates an example CPS with the presence of attacks. Three schemes are defined for modeling attacks. In Scheme-A, the attacker injects malicious data into the communication channel between a sensor and the controller. In Scheme-B, the attacker compromises controllers, and in Scheme-C, there is an attack that is performed in a coordinated way [19].

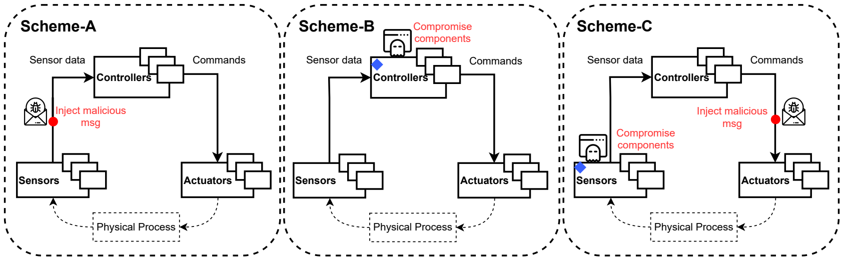


Figure 1.2: The controllers, sensors, actuators, and physical processes are defined as actors in Timed Rebeca model. Scheme-A, Scheme-B, and Scheme-C in Timed Rebeca model are used for security analysis of CPS applications (adapted from [19]). The red circles show attacks on communication and the blue diamonds indicate attacks on components.

We write the correctness properties from system security requirements and feed the Timed Rebeca model augmented with attacks and the property file to the Rebeca model checking tool (Afra) [12, 13, 14] in order to evaluate the system tolerance against the attacks. We check the counter-examples generated by Afra to identify the trace of events leading to a failure.

To build the Tiny Digital Twin, the state space is generated where the attacks are not activated in the Timed Rebeca model. We use our *ltscast* tool [20] to map the state space to the input format of *ltsconvert* tool of mCRL2 [21]. Then, the Tiny Digital Twin is built using *ltsconvert* tool.

Transitions in the state space may be labeled with actions that are not visible to the monitor (and the controller). These actions are typically called non-observable actions. Therefore, these transitions are abstracted away in the Tiny Digital Twin. If we do not abstract the non-observable actions, the monitoring process may become complex and time-consuming. This is due to the fact that the monitor would need to perform a look-ahead search on multiple branches, which can slow down the monitoring process. In general, the continuous behavior of physical components is expressed using differential equations like in Hybrid Automata [22]. In our approach, we abstract the continuous behavior and only model the discrete jump transitions among the states (states are called control modes in hybrid automata). We model the progress of time in each state in Timed Rebeca.

We focus on the software aspect of cyber-physical systems while maintaining an abstract model of the physical world (i.e., Tiny Digital Twin). The model of the physical world is abstract, with particular attention to its interface to the software. Including all details of the dynamics of the system is not necessary for our purpose. For example, in one of our case study (Temperature Control System), we simplify the heating and cooling dynamics by considering the average temperature rate instead of modeling heating and cooling in detail.

Stage_3: Runtime Monitoring. In the third stage, the monitor (implemented in LF) uses the Tiny Digital Twin of the system to detect cyberattacks at runtime [23]. During operational phase of the system, the Tiny Digital Twin is used within a monitor to detect cyberattacks on sensor data and control commands, and identify compromised components such as controllers. The monitor is strategically positioned between the control part and the sensor and actuator components in CPS applications. It observes the visible inputs and outputs of the controllers, traverses state transitions in the Tiny Digital Twin, and detects any misbehavior occurring during system operation. To protect the system against attacks and prevent damage, the monitor drops control commands that are not consistent with the state transitions in the Tiny Digital Twin. Using the Tiny Digital Twin, and the knowledge of the correct and secure functionality of the system, enables the monitor to validate the sequence of actions and the completion time of processes. The monitor is developed using LF language and has the same functionality in different CPS applications. It adjusts the input/output ports based on the number of sensors and actuators of the system.

We demonstrate the details of the methodology to detect attacks using Pneumatic Control System (PCS), Temperature Control System (TCS), and Secure Water Treatment System (SWaT) [23, 20, 24]. The PCS and SWaT systems are distributed control systems whereas TCS is a centralized control system. Aligning logical and physical time, enables us to perform the monitoring at runtime. Relying only on the logical times defined in the model is not a realistic assumption at runtime. We model both periodic and trigger sensors, which are two different types of sensors used in PCS and TCS. The use of different sensor types in PCS and TCS systems highlights the importance of adapting the modeling approach to the specific characteristics of each system. We show how we can model the impact of the environment on the TCS system functionality by using nondeterministic assignment for state variables. We highlight the multiple incoming connections and the utilization of priorities for events during the development of the SWaT case study in LF.

An Example Using CRYSTAL [2] Here, the details of each stage of the CRYSTAL framework are explained in a simplified version of a Pneumatic Control System (PCS) [24]. Figure 1.3 illustrates the control system where it regulates the movement of two cylinders in multiple directions. Each cylinder is controlled by a dedicated controller to regulate the movement in either left-right or up-down directions. The desired sequence of movements of the cylinders is as follows: (1) CylinderB moves down (picks up a particle), (2) CylinderB moves up, (3) CylinderA moves right (pushes CylinderB to the right), (4) CylinderB moves down (leaves the particle) and (5) CylinderB moves up, (6) CylinderA moves left. We assume that ControllerA starts its linear motion from the left at the top of location X.

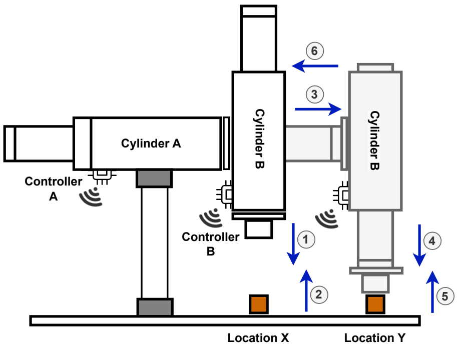


Figure 1.3: PCS with two cylinders (adapted from [25]). The cylinders work together to pick up a particle from location X and move it to location Y.

Timed Rebeca model and LF code. We model the PCS example in Timed Rebeca as depicted in Listing 1. Each component of the system, plus an attacker, is modeled as a reactive class in Timed Rebeca. The reactive class attacker models performing injection on communication channels in the system (line 36). The controllers also receive the information about the other cylinder movements using the message server getctl (line 16). The controllers decide whether to move the cylinder based on the current status and desired movement (lines 5 to 15). The controllers send the motion commands 1 or -1 to regulate the movements (lines 10 and 15). The sensors in this system are the trigger sensors and simply serve as intermediaries between the cylinders

[2]The detailed description of the example is presented in Paper E [24]. In this section, we use a simplified version of PCS to help explain our methodology.

and controllers, reporting location information (line 18) when cylinders touch initial location or target location. The time of the movement for each cylinder is modeled using `after` primitive (line 30).

We use the mapping between Timed Rebeca and Lingua Franca presented in [15] and write a LF code (target c++ language) for the PCS example. The code is presented in Listing 2. To simulate the attacks, we modify the reactions in the reactors. This way, the reactors behave as compromised components and respectively send false sensor data and faulty control commands on the output ports. In addition, the reactor attacker is defined to inject false messages into the channel between the controllers. We can execute the LF compiler and generate an executable file.

Attack models and safety properties. The Timed Rebeca model of the system is augmented with different types of attacks that can be launched to test the resilience of the system. The attack scenarios that are modeled in this case include compromised sensors, compromised cylinders, and injection attacks on the communication channels between controllers as shown in Figure 1.4. In addition, the combined attacks are performed by involving injection attack with the compromised version of sensors and cylinders to perform complex coordinated attacks.

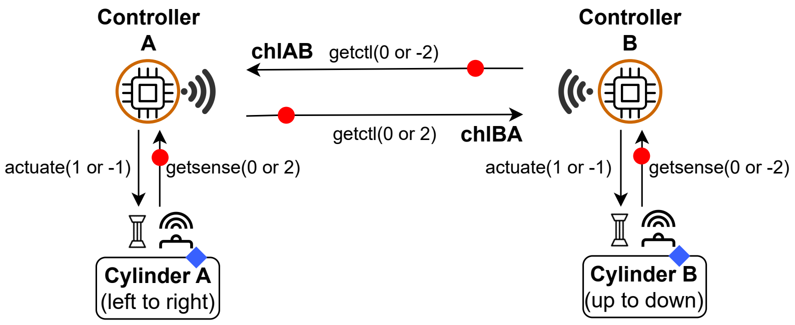


Figure 1.4: The messages are transmitted between controllers in order to regulate the movements. The possible attack points for performing attack scenarios are depicted with red circles and blue diamonds to show attacks on communications or components, respectively.

In order to test for possible complex attack scenarios, we must generate combinations of different values for both the input parameters of the attacker and the compromised components, and verify the model for each combination. To automate this process, we develop a Python script for generating input

```
//env variables  ...
reactiveclass ControllerA(5){ ...
    //locBisUP:true means that CylinderB is up
    msgsrv getsense(int locA) {
        if(locBisUP) {
            if(locAisLeft) {
                if(locA == cylinderAEndloc) {
                    cntlB.getctl(locA); locAisLeft = false;
                locBisUP = false;
                } else { cylA.actuate(1);}
            } else if (!locAisLeft) {
                if(locA == 0) {
                    cntlB.getctl(locA); locAisLeft = true;
                locBisUP = false;
                } else { cylA.actuate(-1); } } } }
    msgsrv getctl(int locB){ if (locB == 0) { locBisUP = true; }}
}
reactiveclass SensorA(5){ ...
        msgsrv getloc(int loc) { ControllerA.getsense(loc);}
}
reactiveclass CylinderA(5){...
        CylinderA(boolean compromised, int compTime, int msg){
                loc = 0; motion = 0; self.status();
                if (compromised) {
                self.actuate(msg) after(compTime); }}
        // left to right on x-axis
        msgsrv status() { loc = loc + motion;
            if(loc == 0 || loc == cylinderAEndloc){
                SensorA.getloc(loc);  motion = 0; }
                self.status() after(2); }
        msgsrv actuate(int rate) { motion = rate;}
}
reactiveclass ControllerB(5){...}
reactiveclass SensorB(5){...}
reactiveclass CylinderB(5){...}
reactiveclass Attacker(3){
        //injects false messages in channels between controllers
        knownrebecs{ ControllerA cntlA; ControllerB cntlB;}
        Attacker(boolean inj, int channel, int msg, int attktime) {
         if(inj){  if (channel == 1) { self.chlBA(msg, attktime);}
                    if (channel == 2) { self.chlAB(msg, attktime); }}}
        msgsrv chlBA(int msg, int attktime){
                cntlA.getctl(msg) after(attktime); }
        msgsrv chlAB(int msg, int attktime){
                cntlB.getctl(msg) after(attktime); }
}
main{
    CylinderA cylA(SensorA):(cAComp,cAComp_time,cAmalMsg);
    ControllerA ControllerA(cylA, ControllerB):();
    SensorA SensorA(ControllerA):(sAComp,sAComp_time,sAmalMsg);
    ... // ControllerB and SensorB instances
    Attacker attacker(ControllerA, ControllerB):
                    (inj_attk, chl, malMsg, attTime);}
```

Listing 1: A part of the Timed Rebeca model augmented with attacks for the PCS example.

```
target Cpp {fast: false, threads: 1};
//loads Tiny Digital Twin
import Monitor.lf;
reactor ControllerA { ...
    if (locBisUP) {
        if (locAisLeft) {
            if(*getctl.get() == 2){ getctlB.set(*getctl.get());
                locAisLeft = false; locBisUP = false; }
            else { actuate.set(1); }
        } else if (!locAisLeft) {
            if(*getctl.get() == 0){
                getctlB.set(*getctl.get());
                locAisLeft = true; locBisUP = false;
            } else { actuate.set(-1); } } } =}
    reaction(getctl) {=
        if (*getctl.get() == 0) { locBisUP = true; } =}
}
reactor SensorA {...
        if(compromised && elapsed_secs
                == std::chrono::seconds(compTime)){
            out.set(msg);
        } else { out.set(sensedValue.get()); }
    =}
}
reactor CylinderA {...
        if(compromised && elapsed_secs
                == std::chrono::seconds(compTime))
        { motion = msg;
        } else { motion = *actuate.get(); }
    =}
    timer status(0, 2 sec);
    reaction(status) -> getloc {=
        loc = loc + motion;
        if(loc == 0 || loc == 2){ getloc.set(loc); motion = 0; } =}
}
reactor ControllerB { ... }
reactor SensorB { ... }
reactor CylinderB { ... }
reactor Attacker { ... } //injections
main reactor PCS {
    ...
    ControllerA = new ControllerA();
    cylA.getloc -> SensorA.sensedValue;
    SensorA.out -> Monitor.getsenseA after 1 sec;
    ControllerA.actuate -> Monitor.actuateA;
    Monitor.cmd_actuateA -> cylA.actuate;
    Attacker.getctlA -> ControllerA.getctl;
}
```

Listing 2: A part of LF code for the PCS example.

values and collecting verification results. This approach is similar in nature to the automated verification technique that uses symbolic modeling and constraint solving. The complete model of the system and the written python codes are available on GitHub [26].

We define safety properties to catch unsafe and undesirable movements of the cylinders. To specify the properties, we use *assertions*. The Timed Rebeca model for the PCS satisfies all the properties in Table 1.1, when none of the attacks are activated. If the model checker detects that a safety property is not satisfied (when the compromised components or injection attacks are activated), it provides the modeler with a counter-example that outlines the sequence of events leading to the violation. This sequence of events can be used to determine the steps of the successful attack scenario.

Table 1.1: The counter-examples for safety properties

#	safety property	counter-example
1	!((motionR&&motionU)||(motionL&&motionD))	$S67 \xrightarrow{controllerb.getsense[0]} S65,...,S70 \xrightarrow{cyla.actuate[1]} S83$
2	!(locXa_outRange||locXb_outRange)	$S67 \xrightarrow{controllerb.getsense[0]} S65,...,S74 \xrightarrow{controllera.getsense[2]} S84$
cyla: CylinderA, cylb: CylinderB, actuate[1]: moves right/down, actuate[-1]: moves left/up		

Table 1.1 shows the safety properties along with two example counter-examples. The properties are defined using the values of the variables loc and motion for two cylinders in the PCS Timed Rebeca model. Property #1 ensures that both cylinders do not move diagonally. In this system, CylinderB cannot move up (motionU) or down (motionD) while CylinderA is moving to the right (motionR) or left (motionL). Property #2 ensures that both CylinderA and CylinderB only have motion between the initial position and the end position in locations X or Y. We checked 3037 attack scenarios on the Timed Rebeca model of PCS (i.e., using Afra model checker) and we found 383 cases where attacks violated safety properties.

Tiny Digital Twin. We create the Tiny Digital Twin for the PCS by providing the *ltsconvert* [21] tool with a list of labels that denote the silent transitions (non-observable actions or tau transitions). Therefore, we developed a tool, *ltscast*, to map the state space created by Afra into the input format of the mCRL2 *ltsconvert* tool. We create the Tiny Digital Twin by abstracting away non-observable actions while preserving trace equivalence [20]. In PCS, the actions getsense, actuate and getctl are observable in the system behavior from the controller point of view, while actions status and getloc are non-observable. The resulting abstract model has 87 states and 120 transitions, while the original state space has 276 states and 439 transitions.

Monitoring and attack detection. We implement the monitor as a reactor in LF (i.e., Monitor.lf) (in Listing 2 line 2). The reactor Monitor is imported to the LF code. The reactor Monitor contains two reactions, one for loading Tiny Digital Twin and another for comparing input data with the transitions in the model.

We consider those compromised components and injection attacks that successfully violate the safety properties at design-time in evaluating the detection capability of the monitor at runtime. In our experiments, we simulate 383 attack scenarios. The scenarios are 355 false sensor data and faulty actuation, 28 injection attacks by defining an attacker, and 281 combined attacks where the injection attacks are combined with sensor data and faulty commands.

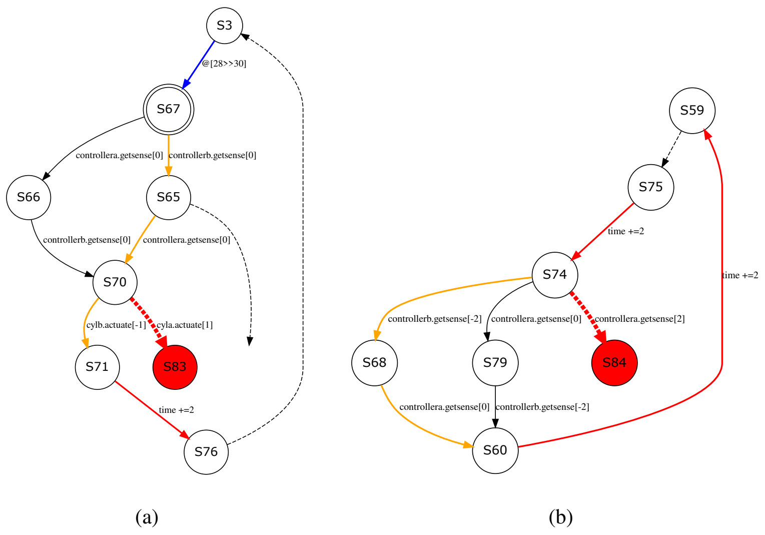


Figure 1.5: The violation of property #1 (a) and property #2 (b) are shown on a subset of the state transitions in the Tiny Digital Twin for the PCS system.

In the counter-example for property #1 (see Table 1.1), as shown in the state transitions in Figure 1.5(a), in state S67, controllerA and controllerB receive the location information of the cylinders through sensor data, i.e., controllerb. getsense[0] and controllera.getsense[0] (state S67 to state S70). At state S70, according to the outgoing transition, the intended action is cylb.actuate[−1] that actuates CylinderB to move downward and pick up the particle. However, at the current state, the action cyla.actuate[1] is transmitted by controllerA to move CylinderA to the right (see the dotted red outgoing transition from S70 to S83). The monitor detects a deviation at state S70 and consequently drops the action cyla.actuate[1]. According to the transition controllera.getsense[0] from S65 to S70, the monitor module keeps state S70 as the current state and proceeds with the system execution.

In the counter-example for property #2, controllerA receives sensor data controllera.getsense[2] at state S74 as shown in Figure 1.5(b). At state S74, CylinderB has been moved down and the location information is transmitted to the controllerB through sensor data controllerb.getsense[-2]. However, at the current state, state S74, incorrect sensor data is injected into the system. The monitor detects the deviation and drops the sensor data controllera.getsense[2]. In this case, the incorrect sensor data can be injected either by an attacker or by a compromised sensor. The monitor (when incorrect sensor data is dropped) proceeds and checks the status of the system execution. This failure can be recovered using a redundant sensor if the original sensor has been compromised.

1.1 Thesis Overview

This thesis is divided into two parts. The first part is a summary of our research, including the background concepts that are used in the thesis (Chapter 2), the problem formulation and our research goals (Chapter 3), the research methods applied in this thesis (Chapter 4), a brief overview of our contributions (Chapter 5), a discussion on the related work (Chapter 6), as well as future work (Chapter 7). The second part is a collection of papers included in this thesis, listed as follows:

Paper A: *On-off attack on a blockchain-based IoT system*, **Fereidoun Moradi**, Ali Sedaghatbaf, Sara Abbaspour Asadollah, Aida Čaušević, and Marjan Sirjani. The 24th IEEE Emerging Technologies and Factory Automation (ETFA 2019), First Workshop on Secure and Trustable Wirelessly Connected Industrial IoT, Zaragoza, Spain, 10th - 13th September 2019.

Abstract: In this paper, we investigate the trust mechanism of Lightweight Scalable BlockChain (LSB), which is a Blockchain specifically designed for Internet of Things networks, to show that a malicious participant in a

Blockchain architecture has the possibility to pursue an On-Off attack and downgrade the integrity of the distributed ledger. We choose a remote software update process as an instance to represent this violation. Finally, using the actor-based language Rebeca, we provide a model of a system under attack and verify the described attack scenario.

Paper contributions: The main contribution of this paper is modeling the On-Off attack scenario using the Timed Rebeca and verifying the trust mechanism of Lightweight Scalable Block Chain (LSB) by performing model checking. Furthermore, the integrity property of LSB is defined as an assertion to indicate in which state of the LSB protocol the property is violated.
My role: I was the main driver of the work. The co-authors contributed with reviews and comments for improving the paper and suggestions on how to accomplish the paper results.

Paper B: *An actor-based approach for security analysis of cyber-physical systems*, **Fereidoun Moradi**, Sara Abbaspour Asadollah, Ali Sedaghatbaf, Aida Čaušević, Marjan Sirjani, and Carolyn Talcott. The 25th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2020), Vienna, Austria, 2nd-3th September 2020.

Abstract: Cyber-Physical Systems are networks of interconnected computing, networking, and physical devices. Communicating through a network makes these systems vulnerable to possible malicious attacks. These systems may play a crucial role in controlling critical infrastructures and their security is of paramount importance. Formal modeling and verification can be effectively used for evaluating secure designs, particularly at the architecture level of complex systems. In this work, we present an actor-based approach for security analysis of cyber-physical systems at the design phase. We use Timed Rebeca, an actor-based modeling language, to model the behavior of components and potential attacks, and verify the security properties using the Rebeca model checking tool. We employ the STRIDE model as a reference for classifying the attacks. To demonstrate the applicability of our approach, we use a Secure Water Treatment (SWaT) system as a case study. We analyze the architecture of the SWaT system using three different attack schemes in which various parts of the system network and physical devices are compromised. In the end, we identify single and combined attack scenarios that violate security properties.

Paper contributions: In this paper, three attack modeling schemes are

defined and employed to discover possible vulnerabilities in communication and components of cyber-physical systems. A formal approach is proposed to evaluate the system security using model checking. In addition, our evaluation of a Secure Water Treatment (SWaT) system resulted in finding several attack scenarios that violate security properties.
My role: I was the main driver of the work. The co-authors reviewed the work and helped in finding the correct terminologies and structure for writing the paper. They pointed out the possible extension of the work and enriched the main idea.

Paper C: *Monitoring cyber-physical systems using a tiny twin to prevent cyber-attacks*, **Fereidoun Moradi**, Maryam Bagheri, Hanieh Rahmati, Hamed Yazdi, Sara Abbaspour Asadollah, and Marjan Sirjani. The 28th International Symposium on Model Checking Software (SPIN 2022), Virtual Event, May 21, 2022.

Abstract: We propose a method for detecting attacks on sensor data and control commands transmitted within cyber-physical systems. The method involves developing a monitor that uses an abstract behavioral model, called the Tiny Digital Twin, to detect false sensor data and faulty control commands. The Tiny Digital Twin is a state transition system that represents the observable behavior of the system from the monitor (and controllers) point of view. At runtime, the monitor observes the sensor data and control commands and checks whether they are consistent with the state transitions in the Tiny Digital Twin. If an inconsistency is detected, the monitor produces an alarm. To build the Tiny Digital Twin, we use the Rebeca modeling language to model the system components and physical processes. We then use the Afra model checker to generate the state space, which is automatically reduced to keep the observable behavior of the system and preserve trace equivalence. We demonstrate the effectiveness of our method by applying it to a Temperature Control System (TCS) case study. The evaluation shows that the approach is capable of detecting attacks and enhancing Cyber-Physical System security.

Paper contributions: This work has two contributions: first, we create the Tiny Digital Twin by abstracting the behavioral model of the system, second, we propose a monitoring algorithm that uses the Tiny Digital Twin to detect disturbances against the system. We implement a monitor in Lingua Franca (LF) and test it on an example system.
My role: I was the main author of the paper. The second author helped

in examining the abstraction techniques for reducing the state space and presenting the theories behind the work. The third and fourth co-authors helped in writing several tiny examples in Timed Rebeca and developing the monitor. The two last co-authors helped to improve the main idea, writing and revising the paper, and choosing the venue for submission.

Paper D: *Tiny Twins for Detecting Cyber-Attacks at Runtime using Concise Rebeca Time Transition System*, **Fereidoun Moradi**, Bahman Pourvatan, Sara Abbaspour Asadollah, and Marjan Sirjani. Journal of Parallel and Distributed Computing (JPDC), page 104780, 2023.

Abstract: In this paper, we present a formal foundation for our approach to mapping the state space of the Timed Rebeca model into a Labeled Transition System (LTS), abstracting the model, and creating a Tiny Digital Twin. We achieve this by defining a Concise Rebeca Timed Transition System (CRTTS) and implementing an *ltscast* tool to convert CRTTS into an LTS. To abstract away non-observable actions from the LTS while preserving trace equivalence between the original model and its abstracted version, we use the mCRL2 *ltsconvert* tool. A monitor is designed where it uses the Tiny Digital Twin to check the consistency of the actual system behavior at runtime. The monitor deals with physical time in the real world based on physical clocks, whereas time in the Timed Rebeca model and subsequently in the Tiny Digital Twin is represented as logical time. To synchronize logical time and physical time, we develop the monitor using the coordination language Lingua Franca (LF), which aligns these two timelines at runtime. LF employs a scheduler that monitors the local clock of each actor and delays processing the message until its measurement of physical time exceeds a threshold. We demonstrate the applicability of our approach in detecting attacks using a Temperature Control System (TCS) case study.

Paper contributions: The main contributions are the definition of theoretical concepts and developing the *ltscast* tool for mapping the state space into a format that is an input of the *ltsconvert* tool.
My role: I was the main author of the paper. The second author helped me to understand the logic for describing the mapping techniques and how to properly specify and write the mathematical equations. The other co-authors helped to improve the main idea behind the work, and they revised and improved the manuscript for submission.

Paper E: *CRYSTAL Framework: Cybersecurity Assurance for Cyber-Physical Systems*, **Fereidoun Moradi**, Sara Abbaspour Asadollah, Bahman Pourvatan, Zahra Moezkarimi, and Marjan Sirjani. Journal of Logical and Algebraic Methods in Programming (JLAMP), 2024.

Abstract: We propose CRYSTAL framework for automated cybersecurity assurance of cyber-physical systems (CPS) at design time and runtime. We build attack models and apply formal verification to recognize potential attacks that may lead to security violations. We focus on both communication and computation in designing the attack models. Using CRYSTAL, we are able to systematically model and check complex coordinated attacks. The cornerstone of CRYSTAL is its architecture based on the MAPE-K[3] feedback loop [11] where we have components to monitor the system, analyze the behavior of the system, plan accordingly, and actuate necessary actions. In this paper we discuss the applicability of CRYSTAL in security analysis and attack detection for different case studies, Pneumatic Control System (PCS), Temperature Control System (TCS), and Secure Water Treatment System (SWaT). We provide a detailed description of the framework and explain how it works in different cases.

Paper contributions: The CRYSTAL is designed to complement an industrial cybersecurity program. The CRYSTAL framework comprises modeling and programming for constructing cyber-physical systems, security analysis through model checking, and runtime monitoring. It provides the industrial system development with an opportunity to identify areas where existing processes can be strengthened.
My role: I was the main author of the paper. The co-authors helped to improve and precisely revise the case studies in the manuscript for submission.

[3]Monitor, Analyze, Plan, Execute - Knowledeg-base

Chapter 2

Background

In this chapter, we present the background required by the current research, helping in the understanding of its content.

2.1 Security Terminology

Vulnerabilities, Threats and Attacks. The security of a system can be compromised by *vulnerabilities*, which refer to security flaws in the design or software of the system that create potential security violations [27]. A *threat* is any malicious behavior that attempts to gain access to the system without permission, while an *attack* is an action that exploits the vulnerabilities [28, 29].

Modeling Attacks. The attack models show the system from the attacker's point of view. By designing attack models, we can observe the behavior of the system under attack (i.e., attack scenarios) and address possible vulnerabilities that could be exploited by attackers [30].

STRIDE. Microsoft STRIDE [31] threat modeling methodology aims to ensure that the design of the system meets the security requirements. The STRIDE stands for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. The STRIDE supports recognizing vulnerabilities, understanding attacks, and proposing mitigation strategies. It shows which aspects of the security of the system (confidentiality, integrity, and availability) can be compromised.

2.2 Model Checking and Abstracting Model

Following are some basic concepts related to this thesis, including system models, model checking, abstraction, and system monitoring.
Model Checking. Model checking is a formal method that examines all possible behaviors of a system model to determine whether a specified property is satisfied [32]. Model checking algorithms are embedded in tools that provide integrated environments to use specification languages for modeling and verification purposes. A model checker tool receives two inputs: a model of the behavior of the system, and a set of properties [33, 32]. The model checking algorithm generates a state space, i.e. state transition diagram, that indicates the changes in system behavior. The increase in size and complexity of the system may lead to the explosion of state space. In case of a property violation, the model checker tool produces a *counter-example* showing the path of the events that lead to the violation.
Abstracting Models. In the context of system modeling and analysis, there are different purposes for which system models are built [34]. Modeling the behavior of the system is a useful way of identifying possible faults. The behavioral model represents the behavior of the system based on the system specification and the requirements. However, sometimes the model of a system can be complex due to the involvement of various entities and features. Model abstraction techniques allow significantly reduce the complexity of the analyzed systems [35]. For example, homomorphism techniques map concrete states and action labels to their abstract versions [35].

2.3 System Monitoring

Monitoring is a technique to check the correctness of execution traces of a system [36]. It verifies certain properties over the system operation and involves analysis techniques to observe the internal behaviour of a system while the system interacts with other external entities. In addition to checking the correctness of the system function during the operational phase, the *monitor* can be used to detect disturbances and provide details to take appropriate action. In the literature, the approach of monitoring along with adaptation mechanism is referred to as the MAPE-K[1] feedback loop [11] where verified model of the system (called Model@runtime) is stored as the knowledge component [37] in the feedback control loop. We need to generate an

[1]The acronym MAPE-K stands for **M**onitor, **A**nalyze, **P**lan, **E**xecute, plus **K**nowledge.

executable component(s) (monitors) for a system that can be used to check the desirable properties during the system operation.

In order to construct an executable monitor, various types of programming languages are available on different platforms that meet the system requirements and are compatible with the developed system. However, it can be more convenient to use specification languages that are directly converted to respective executable code.

We use Lingua Franca (LF) [16] to build an executable model of the system. LF is a programming language based on the Reactor model of computation [17] for building CPSs. In LF, a target language such as C or C++ can be chosen for writing the body of reactors. Reactors are very close to Rebeca in syntax and semantics as shown in [15], and this enables us to effectively generate an executable target code from Timed Rebeca models.

2.4 Timed Rebeca

Rebeca (Reactive Object Language) [38] is an actor-based language for modeling and formal verification of concurrent and distributed systems. An actor, called *rebec* (reactive Object), is an instance of a *reactive class*. Rebecs communicate via asynchronous message passing, which is non-blocking for both sender and receiver. Timed Rebeca, as an extension of Rebeca, has a notion of logical time. The logical time is local times of actors synchronized among all actors, that can be seen as a global time. Each actor has a set of variables that stores values, a set of methods (called *message servers*) and a message bag to store the received messages along with their arrival times and their deadlines. The actor takes a message with the least arrival time from its bag and executes the corresponding *message server*. The actor can change the values of its variables and send messages to its *known actors* while executing a message server. In Timed Rebeca, the primitives *delay* and *after* are used to model the progress of time while executing a message server.

Timed Rebeca is supported by Afra model checker tool [14]. Afra generates the state space of the Timed Rebeca model, in which states contain the local state of all actors and the logical time, and transitions represent three types of possible actions including taking a message from the message bag, executing the corresponding message server of the enabled actor, and progressing the logical time of the model. An approach based on a *shift-equivalence relation* is proposed in [13] to make the state space of a Timed Rebeca model bounded. Two states are in the shift-equivalence relation when all the elements of both states have the same value except for the elements related to time (like the current time value, and the time tags on the messages in the queues including deadlines). The elements related to time can be different but they should all have the same difference (shift) in their amount.

2.5 Lingua Franca (LF)

Lingua Franca is a coordination language based on the Reactor model for programming CPSs [16, 39]. A Reactor model is a collection of *reactors* (like rebecs in Rebeca). A reactor has one or more routines that are called *reactions* (like message servers in Rebeca). Reactions define the functionality of the reactor, and have access to a *state* shared with other reactions, but only within the same reactor (similar to Rebeca). Reactors have named (and typed) *ports* that allow them to connect to other reactors. Two reactors can communicate if an *output* port of a reactor is connected to an *input* port of the other reactor. The usage of *ports* establishes a clean separation between the functionality and composition of reactors; a reactor only references its own ports. Reactions are similar to the message handlers in the actor model. Reactions are triggered by discrete events and may also produce them (similar to handling a message and sending a message). An event relates a value to a *tag* that represents the logical time at which the value is present (similar to a time tag for a message). An event produced by one reactor is only observed by other reactors that are connected to the port on which the event is produced. Events arrive at input ports, and reactions produce events via output ports.

In LF, the logical time does not advance during a reaction. A reactor can have one or more *timers*. Timers are like ports that can trigger reactions. A timer has the form *timer name(offset, period)* that once triggers at the time shown by *offset* (if *offset* is zero, then the timer triggers at the start time of the execution), and then triggers periodically according to the *period*. LF has a built-in type for specifying time intervals. A time interval consists of an integer value accompanied by a time unit (e.g., *sec* for seconds or *msec* for milliseconds). Timers are used for specifying periodic tasks, which are very common in embedded computing and CPSs. Each LF code contains a *main reactor* that is an entry point for the execution of the code. The mapping of Timed Rebeca to Lingua Franca and reverse, including the timing features, is a natural mapping that is discussed in [15, 18].

Chapter 3

Research Problem

In this chapter, we describe the challenges and overall goal and present a summary of sub-goals to meet different aspects of the main research goal.

3.1 Problem Formulation

Cyber-Physical Systems face various threats from cyber-physical attackers. The impacts of these attacks are beyond data loss, as they may damage human and physical processes [40].

Researchers have worked hard to make security rules and plans to keep cyber-physical systems safe from cyberattacks. They mainly focus on stopping these attacks at different levels, using what they call the defense-in-depth strategy [41]. While this is important for protecting systems, we also need to make sure that systems are more resilient to attacks if and when malicious actors gain access. One important step in making these systems more resilient is identifying security vulnerabilities unique to the systems during the design phase. Once the vulnerabilities have been identified, we can take steps to fix the vulnerabilities. This might mean making small changes to the code that controls the system or redesigning parts of the system to make them stronger (e.g., including intrusion detection systems (IDS) or forcing policies and rules). Depending on the criticality of the system, and the extent to which it is vulnerable to attack, advanced system monitoring, adaptation methods, and control strategy may be appropriate. Overall, the following challenges arise while targeting the security of CPS applications: a) The uncertainty in the environment of CPS applications, and errors in components make it hard to keep the whole system secure. b) These systems have different components that work in different ways, which makes it easier for attackers to do complex and coordinated attacks. c) Modeling the behavior of both cyber and physical parts is a challenge because they possess fundamentally different semantics (i.e., discrete and continuous). d) The interactions between cyber components and physical components make it challenging to predict control program behavior at runtime.

Formal methods allow one to specify the design of systems with mathematical models and check the desirable properties. One of the key questions in applying formal verification methods for CPS security is how to take advantage of existing formal mechanisms to automatically enhance CPS security. Thus, the overall goal of this study is: ***"facilitating the security analysis of cyber-physical systems through modeling and formal methods to automatically discover vulnerabilities at the design phase and detect cyberattacks during the operational phase"***.

The following research questions help us in achieving our overall goal:

RQ1: *How can we model cyberattacks to systematically check vulnerabilities?*

To address this research question we employ Timed Rebeca to specify the components of the system and their interactions and build different types of attacks against the security of the system. We systematically augment various malicious changes to the functionality of components and the communications between the components in the model and discover the sequence of actions that violate the system requirements. This research question is aligned with **Sub-Goal 1** that is presented in chapter 3.2.

RQ2: *How to detect cyberattacks at runtime effectively using our design model?*

To address this research question we develop a monitor that uses an abstract behavioral model as a reference model (called a Tiny Digital Twin) to detect cyberattacks. We create the Tiny Digital Twin while removing the non-observable actions (the point of view of the monitor) from the state space and abstracting the model at the right level of equivalency. We use LF language that synchronizes the logical time in the model with the physical time of the system to be able to check the behavior of the system at runtime. We detect disturbances by comparing the actual runtime behavior with the specified behavior in the model. However, the behavioral models of complex systems are usually big and include actions that may be non-observable during monitoring. The abstraction techniques help to remove the non-observable actions from the model and reduce the complexity of the analyzed systems. There are various

fundamental theories for abstracting behavioral models. We use classical abstraction methods such as trace equivalence to reduce the model. This research question is related to **Sub-Goal 2** that is presented in chapter 3.2.
RQ3: *How to assess the applicability of the automated discovery and the runtime monitor in detecting attacks*?

We use three case studies to assess the discovery of vulnerabilities at the design phase and check the successful attacks discovered at runtime. This research question focuses on evaluating the runtime monitor's detection capability with attack scenarios across various CPS applications, aligning with **Sub-Goal 1** and **Sub-Goal 2** that are presented in chapter 3.2.

3.2 Research Goals

According to the main goal, "*facilitating the security analysis of cyber-physical systems through modeling and formal methods to automatically discover vulnerabilities at the design phase and detect cyberattacks during the operational phase*", we define sub-goals below that concern the specific challenges.

- **Sub-Goal 1**: *Create attack models of importance/relevance for the system security objectives*

  We increase the safety and security of the system by building a consistent model and checking the behavior of the system against the respective requirements at the design phase. This step helps to ensure that the system is designed to meet the necessary security requirements. The actor models provide a natural way to represent the various entities and their interactions within the system. Rebeca actor-based modeling language is well-suited for modeling distributed and concurrent systems, and CPS applications often fall into this category. The characteristics of CPS highlight its combination of physical and computational entities, requiring timely interaction, encompassing diverse components, and emphasis on robust design. Using the features of Timed Rebeca, such as message passing, timing features, and non-deterministic assignment, we model the behavior of the system and its environment. In order to conduct a security analysis of the system, various attack scenarios are modeled. The architecture of the system is analyzed using different attack schemes in which various parts of the system network and devices are modeled as compromised versions. These attack scenarios are launched individually or in combination to determine whether the security properties of the system are still being preserved. If a security violation occurs during the analysis, it is important to track the sequence of events leading to the successful violation. This information is used to strengthen the system's security and prevent potential damage.

- **Sub-Goal 2**: *Develop a runtime monitor to check the system behavior at runtime using the Tiny Digital Twin as a reference model*

  It is important to consider changes in system behavior during operation. To achieve this, it is necessary to compare the order and timing of the actions of the actual system with the transitions in the behavioral model. We develop a monitor that uses a reference model (called a Tiny Digital Twin) to observe the actions and detect inconsistencies at runtime. We use Lengua Franca (LF) language to synchronize the logical time in the model with the physical time and develop a runtime monitor to be used in actual systems. We use the similarities between Rebeca actor-based modeling language and Lengua Franca (LF) language to transform the design models to executable codes.

  The behavioral models of complex systems typically has large state spaces, which can make model-based runtime analysis challenging. The Tiny Digital Twin is derived from the state space by abstracting away non-observable actions. Abstraction techniques can help to hide unimportant details and preserve important features for security analysis. The main idea is to have the right level of equivalency to reduce the complexity of the model and the monitoring. In order to abstract the behavioral model and create the Tiny Digital Twin, we map the state space generated by the model checker to a format to be used in action-based abstraction techniques. We assess the capability of our methodology in revealing vulnerabilities and detecting malicious behavior at runtime in various case studies with different levels of complexity and functional objectives. The results of the security analysis show the strengths and limitations of the method.

Chapter 4

Research Methodology

An overview of the research process that is used for our research work is illustrated in Figure 4.1. The core of our research methodology includes four steps adopted from Holz et al. [42] In **Step_1**, we start to define the particular research goals. This step is the initial point for defining the research goals that are related to the real-world problems, (e.g. Cyber-Physical System security and its consequences), the state-of-the-practice (new technologies and tools), and the state-of-the-art literature (scientific research).

In order to formulate a research goal, critical analysis of relevant literature and practice is carried out based on the method presented in [43]. This gives confidence about the defined goals matched with the existing body of knowledge. The formulation of the research goals is not a linear process, maybe they are refined and changed continuously until we figure out the final suitable version of the goals.

In **Step_2**, we propose a solution that addresses an identified research goal. We apply established research methods such as case studies, for developing the solutions. Then, in **Step_3,** the feasibility of the solution is tested in the context, for which we use the proof of concept method (i.e., applying formal methods to verify the correctness of the system design) [42]. The **Step_4** presents the validation process of the research results. The main goal of the validation phase is to assess whether the research results are applicable to the earlier identified real-world problems in the initial step of the research process. This step is performed in close cooperation with industry by applying the proof-by-demonstration [42] research method where both researchers and engineers evaluate the research results. This is an iterative development process where the goals and requirements are changed to meet the desired outcome.

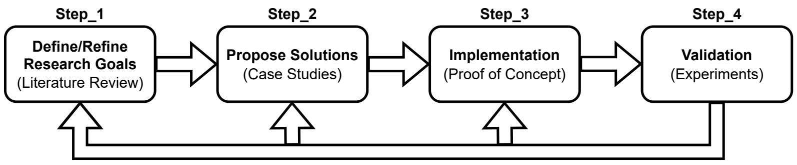


Figure 4.1: Research Methodology

Chapter 5

Thesis Contributions

The following concepts and tools have been completed for the Ph.D. thesis. The four main contributions indicate how they cover the entire proposed plan.

Contribution 1: We present how to model the components of a CPS as actors in Timed Rebeca and define interactions between the components as messages passed between the actors. We define security and safety properties as assertions and provide a method for security analysis using model checking. We categorize attack types in three *attack modeling* schemes considering the STRIDE threat modeling for augmenting attacks in Timed Rebeca models.

Contribution 2: We propose a *monitoring algorithm* to find inconsistencies at runtime. We implement a *monitor* in Lingua Franca (LF) language and simulate several systems under attacks to check the detection capability of the monitor. We present a mapping technique to generate executable codes from Timed Rebeca models.

Contribution 3: We create a *Tiny Digital Twin*; for that, we develop *ltscast* tool to map the state space generated by Afra model checker into a format that can be the input of mCRL2 *ltsconvert* tool, and then we abstract it using the reduction techniques of the tool. We provide a formal foundation for the mapping.

Contribution 4: We present details of our methodology by proposing *CRYSTAL framework*. The capability of the CRYSTAL is shown in three different *case studies*. These case studies highlight the importance of adapting our modeling approach and methodology to the specific characteristics of each system such as centralized and distributed control systems, including periodic and trigger-based sensors, or multiple connections to the control systems.

5.1 Contributions and Goals Relationship

We present the relationship between the contributions of the papers and identified research goals that are included in the Ph.D. thesis as shown in Table 5.1.

Table 5.1: Relationship between Contributions, Papers and Goals.

	C1	**C2**	**C3**	**C4**
Sub-Goal 1	✓			✓
Sub-Goal 2		✓	✓	✓
Papers	**A&B**	**C**	**D**	**E**

The *attack models* are expected in this study. The components of the system and their interactions are defined in the behavioral model. Three attack modeling schemes are defined in **Contribution 1** where attacks on components and communications are augmented in the system model. The outcome of analyzing system design using attack models fulfills **Sub-Goal 1**.

Developing a detection system accomplishes **Sub-Goal 2**, where the *monitor* is responsible for checking the sequence of events using the *Tiny Digital Twin* and distinguishing between the specified behavior and malicious actions as described in **Contribution 2**. The creation of the *Tiny Digital Twin* is supported by a toolset. The theory for mapping the state space to the input format of the mCRL2 and driving the Tiny Digital Twin using the *ltsconvert* tool is presented in **Contribution 3**.

In **Contribution 4**, three *case studies* have been selected to show the feasibility of the methodology. They present the difficulties and limitations of the security analysis in various systems. They validate the outputs from the developed tools that are considered in **Sub-Goal 1** and **Sub-Goal 2**.

Chapter 6

Related Work and Comparison

Modeling attacks: Attack graphs can serve as a basis for detection and security analysis [44]. Byres et al. [45] investigate vulnerabilities in Modbus-based SCADA systems using attack trees. The authors present attack trees for gaining access to the SCADA system and provide an estimated level of technical difficulty, the severity of impact, and underlying critical vulnerabilities for the possible goals of an attacker. Yan et al. [46] develop an attack graph to identify possible intrusion scenarios, and propose an intrusion detection system. Wasicek et al. [47] propose an aspect-oriented technique to model attacks against cyber-physical systems. The authors in [47] use Ptolemy [48] as the modeling and simulation framework and demonstrate the practicality of their technique through modeling four types of attacks (fuzz attack, interruption, man-in-the-middle, and replay attack) on an automotive control system. In [49], Rocchetto and Tippenhauer present a taxonomy of the diverse attacker models for cyber-physical systems security. They employ the attacker models to investigate the impact of single-point cyber attacks on a Secure Water Treatment System (SWaT) [50]. Fritz and Zhang [51] consider cyber-physical systems as discrete-event systems and model them using a variant of Petri nets. They propose a method based on permutation matrices to detect deception attacks. In particular, they can detect attacks by changing the input and output behavior of the system and analyzing its effect on the system behavior. Covert attacks and replay attacks are two kinds of attacks modeled and analyzed in this study.

Discovering vulnerabilities using model checkers: In [52], Kang et al. use Alloy to model a scaled-down version of a Secure Water Treatment System (SWaT) and potential attackers. They can discover the undetected attacks which cause safety failure (e.g., water tank overflow). They compare the actual invariant of the SWaT system and the output state in the Alloy model checker during system operation. Rocchetto and Tippenhauer [53] use the ASLan++ tool for modeling the physical layer interactions and the CL-AtSe tool for analyzing the state space. They succeed to find eight attack scenarios on SWaT. Nigam and Talcott [54] use Maude [55] to automate security analysis of the protocols in Industry 4.0 applications. They formalize networked sets of devices and a symbolic intruder model in rewriting logic. Hailesellasie and Hasan [56] consider verification of the PLC network within an industrial control system by creating graphs of the potentially compromised PLC program and a trusted version of the program. They create a formal model of both programs in UPPAAL, then translate the model to attribute graphs. The matching comparison of the graphs provides guarantees that the system has not been compromised. Their approach is demonstrated in a case study of an industrial water level control system.

Security analysis frameworks: Feng et al. [57] design a framework to systematically generate rules from information contained within the operational logs of industrial control systems. Such rules are then selected by system engineers to generate an invariant-based intrusion detection system. Winnicki et al. [58] propose an approach to discover the behavior of cyber-physical systems with probing. They slightly change the system behavior and observe how the controls react to take the system back to its normal state. They detect attacks by comparing the normal changes and the abnormal alteration in the process. In [59], a model checking-based framework called ForeSee is proposed where it evaluates IoT system security. It builds a multi-layer graph by simultaneously modeling all of the essential components of the system, including the physical environment, devices, communication protocols, and applications. The SPIN model checker [60] can then analyze the generated graph to validate system security properties or generate attack paths if there are any violations. Green et al. [61] provide two practical examples based on a Man-In-The-Middle scenario, demonstrating the types of information an attacker needs to obtain in order to manipulate the process of the system and bypass the detection system. The authors in [62] propose a method using SysML-Sec, a modeling system based on UML, for working with large code bases. The authors suggest creating models in SysML and iteratively refining the behavior of the model by adding more behavioral and security properties using SysML-Sec. The model can be automatically translated to pi-calculus by use of the TTool [63] and formally analyzed by ProVerif.

Runtime monitoring for detecting cyberattacks: Lanotte et al. [64] propose a formal approach based on runtime enforcement to ensure specification compliance in networks of controllers. They define a synthesis algorithm with respect to Ligatti et al.'s edit automata [65]. The algorithm takes an alphabet of observable actions and a timed correctness property and returns an edit automaton as an enforcer. In their work, the enforcers are synthesized regarding the deterministic behavior of the controllers. The network of enforcers preserves weak bisimilarity equivalence in relation to the networks of controllers. In [66], authors propose adaptive security policies at runtime. They use ProB model checker [67] to automatically detect the root cause of security violations. They check design models against security constraints at runtime. The authors in [68] propose monitors expressed as Quantified Event Automata (QEAs) [69] to detect injection attacks against a system. QEAs represent parametric specifications to be checked at runtime. The monitors support event duplication that provides more protection against attacks. They validate Java implementations of the monitors using attack examples on the system [70].

Modeling and verification of cyber-physical systems: Hybrid automata [71] provide formal modeling and analysis for cyber-physical system focusing on the interaction between the discrete and continuous parts. UPPAAL SMC [72] provides statistical model checking for stochastic hybrid systems. In UPPAAL SMC, each component of the system is described as an automaton where integer variables (usually called clocks) can be defined by simple constants, or by general expressions involving clocks, allowing clock valuations to be defined by ordinary differential equations [73]. MODEST toolset [74] contains several analysis backends which can be used for the design and the formal analysis of stochastic hybrid automata. In [75], Lanotte et al. use the safety model checker *prohver* (a tool within MODEST) to verify the behavior of a CPS application. They analyze three attacks targeting sensors and/or actuators of the system by compromising either the corresponding physical device or the communication network used by the device. They specify the system with linear formulae to express nondeterministic assignments within a dense interval and use shared actions to synchronize parallel components. In [76], Hybrid

Rebeca is proposed to support the modeling of cyber-physical systems. In Hybrid Rebeca, physical actors are introduced as new computational entities to encapsulate the physical behaviors. In [15], an example of a cyber-physical system is modeled in Timed Rebeca and it is shown how to identify subtle defects in the design. They propose an approach for formal verification of cyber-physical systems and Lingua Franca codes, where they focus on the cyber part and model a faithful interface to the physical part.

Comparison: We use STRIDE threat modeling to systematically construct attack models, whereas others rely on known attacks to create attack profiles (e.g., [49, 77, 61, 51, 47, 46]). This systematic approach ensures we cover a wide range of potential attacks (single attacks and coordinated attacks), providing a robust security analysis. On the other hand, approaches that use attack profiles typically focus on predefined attack scenarios. These profiles may capture some common attacks but might not address the full range of potential attacks. Secondly, we incorporate actors into our approach, allowing us to represent the various entities and components within the system, along with their interactions. This approach reduces the semantic gap between the model and the entities in real-world applications, enhancing the accuracy and relevance of the system design. In contrast, other approaches (e.g., [52, 53, 56]) prioritize aspects like system architecture or data flow to discover vulnerabilities. They may not explicitly represent the entities and their respective roles. In our approach, the designed model can be mapped into executable code using the natural mapping between Timed Rebeca and Lingua Franca. We also use an abstract behavioral model for runtime monitoring, which is automatically created using formal equivalence techniques. This approach ensures a precise behavioral model, and the detection process becomes efficient as the model represents observable behavior while hiding unimportant details for monitoring. Moreover, the implementation of the methods facilitates the automation of the abstraction process, the generation of executable codes, and runtime monitoring.

In terms of attack detection using Tiny Digital Twin, we compare CRYSTAL framework with some security frameworks such as ForeSee [59], a rule-based framework in [57], PLCDefender [78] and Shade [79]. In [57], a framework is proposed to systematically generate invariant rules from ICS data logs to detect attacks. However, applying rules for detecting attacks is usually difficult because of the generation of too many rules and their false positives. All of these rules are either manually defined or require a large amount of human effort. In CRYSTAL, our Tiny Digital Twin is automatically created and the monitor code is generated easily through the available mapping between Timed Rebeca and LF. This model-based process reduces the human effort to build a detection system and it ensures that the detection system works without false positives. PLCDefender [78] is a remote attestation framework with a physics-based model to preserve the control behavior integrity of PLC, and Shade [79] is a shadow memory technology against control logic tamper attacks. The capabilities of these two frameworks can only be used for specific attacks and cannot comprehensively improve the security of CPS. However, in CRYSTAL, the monitor can be utilized in various systems (e.g., IoT, ICS, or PLC-based systems) and it can detect any cyberattacks that impact the system and show a deviation from the desired behavior. In [59], ForeSee uses a multi-layer graph to evaluate the security of the system. It checks the graph using SPIN to see if there are any deviations from what is expected in the graph and specified security properties. The ForeSee framework focuses on the static analysis of IoT applications and provides some optimization algorithms for reducing the computational complexity of the analysis, while CRYSTAL supports both the design time and runtime phases of the system development to identify vulnerabilities and detect attacks. In [62], a detection method is presented to use SysML-Sec, a UML-based model, for the system with large codes. In their work, TTool [63] is used to translate UML models to pi-calculus. Indeed, UML model may not explicitly represent the entities in the system and the environment. Therefore, the translation and the result pi-calculus may not be accurate enough. In CRYSTAL, we use Timed Rebeca, i.e., a formal language, for modeling entities and the environment in which the language enables us to model the timing and dynamic features of the system.

In CRYSTAL, instead of employing hybrid automata like the work in [72, 75] for modeling and verifying cyber-physical systems, we adopt a similar approach as presented in [15] for modeling cyber-physical systems. We model the software system, which monitors and receives data from the physical processes and sends control commands back to them. By employing this model, we can verify if the software system produces the correct output based on the given requirements when certain data is received. We also model the environment (dynamic of the system) using non-deterministic assignment for the state variables. As an example, to represent a physical component, i.e., the temperature of a physical room, we use an actor where the state variables model different states of the actor. In the Timed Rebeca model, discrete values are assigned to the state variables non-deterministically. We model the changes at certain times (or when an event occurs). We use the *after* construct appended to a send statement to model these changes during the time. Any change (event) in the system is modeled by executing a message server (handling the message).

Chapter 7

Future Work

For extending the CRYTAL framework, there are several possible directions. In the runtime monitoring phase (stage_3), a module can be developed to generate repair actions for the monitor. The module recovers the system after attacks based on the adaptation plans, instead of just terminating the system in case of successful attack detection. The module can use reinforcement learning to automatically generate potential protection strategies. In addition, writing policies and rules within the monitor to catch abnormal behavior in the system is an approach that can come along with the Tiny Digital Twin to improve the capability of the monitor. In the design-time security analysis phase (stage_2), known failures can be discovered based on the insights gained from property violations. This approach can involve the development of a pathfinding method, aimed at identifying the recovery paths within the Tiny Digital Twin, with a focus on achieving the shortest possible recovery times. Moreover, proposing self-healing mechanisms that address unknown failures, such as attacks that are not discovered during the design phase (e.g., stealthy attacks) is favorable. In the modeling and code generation phase (stage_1), manual mapping Timed Rebeca to LF codes may introduce additional errors. For this reason, the development of a code transformation, specifically, a compiler that translates a Timed Rebeca model to an LF code, is appreciated.

Bibliography

[1] D. B. Rawat, C. Bajracharya, and G. Yan, "Towards intelligent transportation cyber-physical systems: Real-time computing and communications perspectives," in *SoutheastCon 2015*, pp. 1–6, IEEE, 2015.

[2] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-cps: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. 11, no. 1, pp. 88–95, 2015.

[3] C. Lv, X. Hu, A. Sangiovanni-Vincentelli, Y. Li, C. M. Martinez, and D. Cao, "Driving-style-based codesign optimization of an automated electric vehicle: a cyber-physical system approach," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 2965–2975, 2018.

[4] kaspersky Lab ICS CERT, "Threat landscape for industrial automation systems," 2019.

[5] P. Bell, "Cyber threat report may 06, 2019," 2019.

[6] E. Kovacs, "Dos attack blamed for u.s. grid disruptions," 2019.

[7] NERC, "Risks posed by firewall firmware vulnerabilities," September 2019.

[8] W. Stallings, L. Brown, M. D. Bauer, and A. K. Bhattacharjee, *Computer security: principles and practice*. Pearson Education, 2012.

[9] D. Gollmann, P. Gurikov, A. Isakov, M. Krotofil, J. Larsen, and A. Winnicki, "Cyber-physical systems security: Experimental analysis of a vinyl acetate monomer plant," in *Proceedings of Cyber-Physical System Security*, pp. 1–12, ACM, 2015.

[10] T. Kulik, B. Dongol, P. G. Larsen, H. D. Macedo, S. Schneider, P. W. Tran-Jørgensen, and J. Woodcock, "A survey of practical formal methods for security," *Formal aspects of computing*, vol. 34, no. 1, pp. 1–39, 2022.

[11] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

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

[13] E. Khamespanah, M. Sirjani, Z. Sabahi-Kaviani, R. Khosravi, and M. Izadi, "Timed Rebeca schedulability and deadlock freedom analysis using bounded floating time transition system," *Sci. Comput. Program.*, vol. 98, pp. 184–204, 2015.

[14] Afra, "An integrated environment for modeling and verifying Rebeca family designs," *[Online; accessed Dec 09, 2022]*, 2022.

[15] M. Sirjani, E. A. Lee, and E. Khamespanah, "Verification of cyberphysical systems," *Mathematics*, vol. 8, no. 7, p. 1068, 2020.

[16] M. Lohstroh, Í. Í. Romeo, A. Goens, P. Derler, J. Castrillon, E. A. Lee, and A. Sangiovanni-Vincentelli, "Reactors: A deterministic model for composable reactive systems," in *Cyber Physical Systems. Model-Based Design*, pp. 59–85, Springer, 2019.

[17] M. Lohstroh, M. Schoeberl, A. Goens, A. Wasicek, C. Gill, M. Sirjani, and E. A. Lee, "Invited: Actors revisited for time-critical systems," in *DAC*, 2019.

[18] M. Sirjani, E. Khamespanah, and E. Lee, "Model checking software in cyberphysical systems," in *COMPSAC 2020*, 2020.

[19] 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.

[20] F. Moradi, B. Pourvatan, S. A. Asadollah, and M. Sirjani, "Tiny twins for detecting cyber-attacks at runtime using concise rebeca time transition system," *Journal of Parallel and Distributed Computing*, vol. 184, p. 104780, 2024.

[21] D. N. Jansen, J. F. Groote, J. J. Keiren, and A. Wijs, "An O (m log n) algorithm for branching bisimilarity on labelled transition systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 3–20, Springer, 2020.

[22] T. A. Henzinger, "The theory of hybrid automata," in *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pp. 278–292, IEEE Computer Society, 1996.

[23] 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 Software*, pp. 24–43, Springer, 2022.

[24] F. Moradi, S. A. Asadollah, B. Pourvatan, Z. Moezkarimi, and M. Sirjani, "Crystal framework: Cybersecurity assurance for cyber-physical systems," *Journal of Logical and Algebraic Methods in Programming*, p. 100965, 2024.

[25] Z. Jakovljevic, V. Lesi, and M. Pajic, "Attacks on distributed sequential control in manufacturing automation," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 775–786, 2020.

[26] "Pneumatic control system case study." `https://github.com/fereidoun-moradi/Reconfigurable-Pneumatic-System`, 2023. [Online; accessed Apr 24, 2023].

[27] D. L. Pipkin, *Information security: protecting the global enterprise*. Prentice-Hall, Inc., 2000.

[28] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions," *Electronics*, vol. 12, no. 6, p. 1333, 2023.

[29] C. Paulsen, "Glossary of key information security terms," tech. rep., National Institute of Standards and Technology, 2018.

[30] S. Paudel, P. Smith, and T. Zseby, "Attack models for advanced persistent threats in smart grid wide area monitoring," in *Proceedings of the 2nd Workshop on Cyber-Physical Security and Resilience in Smart Grids*, pp. 61–66, 2017.

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

[32] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[33] E. M. Clarke, O. Grumberg, and D. E. Long, “Model checking and abstraction,” *ACM transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, no. 5, pp. 1512–1542, 1994.

[34] D. Dori, “Object-process methodology,” in *Encyclopedia of Knowledge Management, Second Edition*, p. 249–287, IGI Global, 2011.

[35] T. Sminia and M. Á. V. Espada, “Modal abstraction and replication of processes with data,”

[36] X. Zheng, C. Julien, R. Podorozhny, F. Cassez, and T. Rakotoarivelo, “Efficient and scalable runtime monitoring for cyber–physical system,” *IEEE Systems Journal*, vol. 12, no. 2, pp. 1667–1678, 2016.

[37] M. Bagheri, M. Sirjani, E. Khamespanah, N. Khakpour, I. Akkaya, A. Movaghar, and E. A. Lee, “Coordinated actor model of self-adaptive track-based traffic control systems,” *Journal of Systems and Software*, vol. 143, pp. 116–139, 2018.

[38] M. Sirjani and M. M. Jaghoori, “Ten years of analyzing actors: Rebeca experience,” in *Formal Modeling: Actors, Open Systems, Biological Systems*, pp. 20–56, Springer, 2011.

[39] M. Lohstroh, C. Menard, S. Bateni, and E. A. Lee, “Toward a lingua franca for deterministic concurrent systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 4, pp. 1–27, 2021.

[40] J. M. Taylor and H. R. Sharif, “Security challenges and methods for protecting critical infrastructure cyber-physical systems,” in *2017 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, pp. 1–6, IEEE, 2017.

[41] X. Ning and J. Jiang, “Defense-in-depth against insider attacks in cyber-physical systems,” *Internet of Things and Cyber-Physical Systems*, vol. 2, pp. 203–211, 2022.

[42] H. J. Holz, A. Applin, B. Haberman, D. Joyce, H. Purchase, and C. Reed, “Research methods in computing: what are they, and how should we teach them?,” *ACM SIGCSE Bulletin*, vol. 38, no. 4, pp. 96–114, 2006.

[43] M. V. Zelkowitz and D. Wallace, "Experimental validation in software engineering," *Information and Software Technology*, vol. 39, no. 11, pp. 735–743, 1997.

[44] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in *Proceedings 15th IEEE Computer Security Foundations Workshop. CSFW-15*, pp. 49–63, IEEE, 2002.

[45] E. J. Byres, M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in scada systems," in *Proceedings of the international infrastructure survivability workshop*, pp. 3–10, Citeseer, 2004.

[46] J. Yan, M. Govindarasu, C.-C. Liu, M. Ni, and U. Vaidya, "Risk assessment framework for power control systems with pmu-based intrusion response system," *Journal of Modern Power Systems and Clean Energy*, vol. 3, no. 3, pp. 321–331, 2015.

[47] A. Wasicek, P. Derler, and E. A. Lee, "Aspect-oriented modeling of attacks in automotive cyber-physical systems," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014.

[48] J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, *Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems*, p. 527–543. Kluwer Academic Publishers, 2001.

[49] M. Rocchetto and N. O. Tippenhauer, "On attacker models and profiles for cyber-physical systems," in *Computer Security–ESORICS 2016: 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part II 21*, pp. 427–449, Springer, 2016.

[50] S. Adepu and A. Mathur, "An investigation into the response of a water treatment system to cyber attacks," in *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 141–148, IEEE, 2016.

[51] R. Fritz and P. Zhang, "Modeling and detection of cyber attacks on discrete event systems," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 285–290, 2018.

[52] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur, "Model-based security analysis of a water treatment system," in *Proceedings of Software Engineering for Smart Cyber-Physical Systems*, pp. 22–28, ACM, 2016.

[53] M. Rocchetto and N. O. Tippenhauer, "Towards formal security analysis of industrial control systems," in *ACM Asia Conference on Computer and Communications Security*, pp. 114–126, ACM, 2017.

[54] V. Nigam and C. Talcott, "Formal security verification of industry 4.0 applications," in *24th IEEE International Conference on Emerging Technologies and Factory Automation*, 2019.

[55] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott, *All About Maude-A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic*, vol. 4350. Springer, 2007.

[56] M. Hailesellasie and S. R. Hasan, "Intrusion detection in plc-based industrial control systems using formal verification approach in conjunction with graphs," *Journal of Hardware and Systems Security*, vol. 2, pp. 1–14, 2018.

[57] C. Feng, V. R. Palleti, A. Mathur, and D. Chana, "A systematic framework to generate invariants for anomaly detection in industrial control systems.," in *NDSS*, 2019.

[58] A. Winnicki, M. Krotofil, and D. Gollmann, "Cyber-physical system discovery: Reverse engineering physical processes," in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, pp. 3–14, 2017.

[59] Z. Fang, H. Fu, T. Gu, Z. Qian, T. Jaeger, P. Hu, and P. Mohapatra, "A model checking-based security analysis framework for iot systems," *High-Confidence Computing*, vol. 1, no. 1, p. 100004, 2021.

[60] G. J. Holzmann, "The model checker spin," *IEEE Transactions on software engineering*, vol. 23, no. 5, pp. 279–295, 1997.

[61] B. Green, M. Krotofil, and A. Abbasi, "On the significance of process comprehension for conducting targeted ics attacks," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, pp. 57–67, 2017.

[62] L. Apvrille, L. Li, and Y. Roudier, "Model-driven engineering for designing safe and secure embedded systems," in *2016 Architecture-Centric Virtual Integration (ACVI)*, pp. 4–7, IEEE, 2016.

[63] A. Enrici, L. Apvrille, and R. Pacalet, "Ttool/diplodocusdf: a uml environment for hardware/software co-design of data-dominated systems-on-chip," *Demonstration at DATE*, 2014.

[64] R. Lanotte, M. Merro, and A. Munteanu, "A process calculus approach to detection and mitigation of plc malware," *Theoretical Computer Science*, vol. 890, pp. 125–146, 2021.

[65] J. Ligatti, L. Bauer, and D. Walker, "Edit automata: Enforcement mechanisms for run-time security policies," *International Journal of Information Security*, vol. 4, no. 1, pp. 2–16, 2005.

[66] H. Loulou, S. Saudrais, H. Soubra, and C. Larouci, "Adapting security policy at runtime for connected autonomous vehicles," in *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 26–31, IEEE, 2016.

[67] M. Leuschel and M. Butler, "Prob: A model checker for b," in *FME 2003: Formal Methods: International Symposium of Formal Methods Europe, Pisa, Italy, September 8-14, 2003. Proceedings*, pp. 855–874, Springer, 2003.

[68] A. Kassem and Y. Falcone, "Detecting fault injection attacks with runtime verification," in *Proceedings of the 3rd ACM Workshop on Software Protection*, pp. 65–76, 2019.

[69] H. Barringer, Y. Falcone, K. Havelund, G. Reger, and D. Rydeheard, "Quantified event automata: Towards expressive and efficient runtime monitors," in *International Symposium on Formal Methods*, pp. 68–84, Springer, 2012.

[70] L. Dureuil, G. Petiot, M.-L. Potet, T.-H. Le, A. Crohen, and P. de Choudens, "Fissc: A fault injection and simulation secure collection," in *International Conference on Computer Safety, Reliability, and Security*, pp. 3–11, Springer, 2016.

[71] T. A. Henzinger, "The theory of hybrid automata," in *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pp. 278–292, IEEE, 1996.

[72] A. David, D. Du, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, and S. Sedwards, "Statistical model checking for stochastic hybrid systems," *arXiv preprint arXiv:1208.3856*, 2012.

[73] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, “Uppaal smc tutorial,” *International journal on software tools for technology transfer*, vol. 17, pp. 397–415, 2015.

[74] E. M. Hahn, A. Hartmanns, H. Hermanns, and J.-P. Katoen, “A compositional modelling and analysis framework for stochastic hybrid systems,” *Formal Methods in System Design*, vol. 43, no. 2, pp. 191–232, 2013.

[75] R. Lanotte, M. Merro, and A. Munteanu, “A modest security analysis of cyber-physical systems: A case study,” in *Formal Techniques for Distributed Objects, Components, and Systems: 38th IFIP WG 6.1 International Conference, FORTE 2018, Held as Part of the 13th International Federated Conference on Distributed Computing Techniques, DisCoTec 2018, Madrid, Spain, June 18-21, 2018, Proceedings 38*, pp. 58–78, Springer, 2018.

[76] I. Jahandideh, F. Ghassemi, and M. Sirjani, “Hybrid rebeca: Modeling and analyzing of cyber-physical systems,” in *Cyber Physical Systems. Model-Based Design: 8th International Workshop, CyPhy 2018, and 14th International Workshop, WESE 2018, Turin, Italy, October 4–5, 2018, Revised Selected Papers 8*, pp. 3–27, Springer, 2019.

[77] S. D. D. Anton, A. P. Lohfink, and H. D. Schotten, “Discussing the feasibility of acoustic sensors for side channel-aided industrial intrusion detection: An essay,” in *Proceedings of the Third Central European Cybersecurity Conference*, pp. 1–4, 2019.

[78] M. Salehi and S. Bayat-Sarmadi, “Plcdefender: Improving remote attestation techniques for plcs using physical model,” *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7372–7379, 2020.

[79] H. Yoo, S. Kalle, J. Smith, and I. Ahmed, “Overshadow plc to detect remote control-logic injection attacks,” in *Detection of Intrusions and Malware, and Vulnerability Assessment: 16th International Conference, DIMVA 2019, Gothenburg, Sweden, June 19–20, 2019, Proceedings 16*, pp. 109–132, Springer, 2019.