

A Formal Approach to Physics-Based Attacks in Cyber-Physical Systems

RUGGERO LANOTTE, Dipartimento di Scienza e Alta Tecnologia, Università dell'Insubria, Italy
MASSIMO MERRO, Dipartimento di Informatica, Università degli Studi di Verona, Italy
ANDREI MUNTEANU, Dipartimento di Informatica, Università degli Studi di Verona, Italy
LUCA VIGANÒ, Department of Informatics, King's College London, UK

We apply formal methods to lay and streamline theoretical foundations to reason about Cyber-Physical Systems (CPSs) and physics-based attacks, i.e., attacks targeting physical devices. We focus on a formal treatment of both integrity and denial of service attacks to sensors and actuators of CPSs, and on the timing aspects of these attacks. Our contributions are fourfold. (1) We define a hybrid process calculus to model both CPSs and physics-based attacks. (2) We formalise a threat model that specifies MITM attacks that can manipulate sensor readings or control commands in order to drive a CPS into an undesired state, and we provide the means to assess attack tolerance/vulnerability with respect to a given attack. (3) We formalise how to estimate the impact of a successful attack on a CPS and investigate possible quantifications of the success chances of an attack. (4) We illustrate our definitions and results by formalising a non-trivial running example in Uppaal SMC, the statistical extension of the Uppaal model checker; we use Uppaal SMC as an automatic tool for carrying out a static security analysis of our running example in isolation and when exposed to three different physics-based attacks with different impacts.

CCS Concepts: • **Security and privacy** → **Formal security models; Logic and verification; Embedded systems security**; • **Theory of computation** → *Timed and hybrid models*;

Additional Key Words and Phrases: Cyber-physical system security, formal methods, theoretical foundations, process calculus.

ACM Reference Format:

Ruggero Lanotte, Massimo Merro, Andrei Munteanu, and Luca Viganò. 2019. A Formal Approach to Physics-Based Attacks in Cyber-Physical Systems. *ACM Trans. Priv. Sec.* 0, 0, Article 0 (2019), 39 pages. <https://doi.org/0000001.0000001>

1 INTRODUCTION

1.1 Context and motivation

Cyber-Physical Systems (CPSs) are integrations of networking and distributed computing systems with physical processes that monitor and control entities in a physical environment, with feedback loops where physical processes affect computations and vice versa. For example, in real-time control systems, a hierarchy of *sensors*, *actuators* and *control components* are connected to control stations.

Authors' addresses: Ruggero Lanotte, Dipartimento di Scienza e Alta Tecnologia, Università dell'Insubria, Como, Italy, ruggero.lanotte@uninsubria.it; Massimo Merro, Dipartimento di Informatica, Università degli Studi di Verona, Verona, Italy, massimo.merro@univr.it; Andrei Munteanu, Dipartimento di Informatica, Università degli Studi di Verona, Verona, Italy, andrei.munteanu@univr.it; Luca Viganò, Department of Informatics, King's College London, London, UK, luca.vigano@kcl.ac.uk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery. 2471-2566/2019/0-ART0 \$15.00
<https://doi.org/0000001.0000001>

In recent years there has been a dramatic increase in the number of attacks to the security of CPSs, e.g., manipulating sensor readings and, in general, influencing physical processes to bring the system into a state desired by the attacker. Some notorious examples are: (i) the *STUXnet* worm, which reprogrammed PLCs of nuclear centrifuges in Iran [36]; (ii) the attack on a sewage treatment facility in Queensland, Australia, which manipulated the SCADA system to release raw sewage into local rivers and parks [53]; (iii) the *BlackEnergy* cyber-attack on the Ukrainian power grid, again compromising the SCADA system [32].

A common aspect of these attacks is that they all compromised *safety critical systems*, i.e., systems whose failures may cause catastrophic consequences. Thus, as stated in [25, 26], the concern for consequences at the physical level puts *CPS security* apart from standard *information security*, and demands for ad hoc solutions to properly address such novel research challenges.

These ad hoc solutions must explicitly take into consideration a number of specific issues of attacks tailored for CPSs. One main critical issue is the *timing of the attack*: the physical state of a system changes continuously over time and, as the system evolves, some states might be more vulnerable to attacks than others [34]. For example, an attack launched when the target state variable reaches a local maximum (or minimum) may have a great impact on the whole system behaviour, whereas the system might be able to tolerate the same attack if launched when that variable is far from its local maximum or minimum [35]. Furthermore, not only the timing of the attack but also the *duration of the attack* is an important parameter to be taken into consideration in order to achieve a successful attack. For example, it may take minutes for a chemical reactor to rupture [55], hours to heat a tank of water or burn out a motor, and days to destroy centrifuges [36].

Much progress has been done in the last years in developing formal approaches to aid the *safety verification* of CPSs (e.g., [5, 19, 20, 29, 48, 49], to name a few). However, there is still a relatively small number of works that use formal methods in the context of the *security analysis* of CPSs (e.g., [2, 9, 11, 12, 46, 50, 57, 59]). In this respect, to the best of our knowledge, a systematic formal approach to study *physics-based attacks*, that is, attacks targeting the physical devices (sensors and actuators) of CPSs, is still to be fully developed. Our paper moves in this direction by relying on a process calculus approach.

1.2 Background

The dynamic behaviour of the *physical plant* of a CPS is often represented by means of a *discrete-time state-space model*¹ consisting of two equations of the form

$$\begin{aligned}x_{k+1} &= Ax_k + Bu_k + w_k \\y_k &= Cx_k + e_k\end{aligned}$$

where $x_k \in \mathbb{R}^n$ is the current (*physical*) state, $u_k \in \mathbb{R}^m$ is the *input* (i.e., the control actions implemented through actuators) and $y_k \in \mathbb{R}^p$ is the *output* (i.e., the measurements from the sensors). The *uncertainty* $w_k \in \mathbb{R}^n$ and the *measurement error* $e_k \in \mathbb{R}^p$ represent perturbation and sensor noise, respectively, and A , B , and C are matrices modelling the dynamics of the physical system. Here, the *next state* x_{k+1} depends on the current state x_k and the corresponding control actions u_k , at the sampling instant $k \in \mathbb{N}$. The state x_k cannot be directly observed: only its measurements y_k can be observed.

The physical plant is supported by a communication network through which the sensor measurements and actuator data are exchanged with controller(s) and supervisor(s) (e.g., IDSs), which are the *cyber* components (also called *logics*) of a CPS.

¹See [61, 62] for a taxonomy of the time-scale models used to represent CPSs.

1.3 Contributions

In this paper, we focus on a formal treatment of both *integrity* and *Denial of Service (DoS)* attacks to *physical devices* (sensors and actuators) of CPSs, paying particular attention to the *timing aspects* of these attacks. The overall goal of the paper is to apply *formal methodologies* to lay *theoretical foundations* to reason about and formally detect attacks to physical devices of CPSs. A straightforward utilisation of these methodologies is for *model-checking* (as, e.g., in [20]) or *monitoring* (as, e.g., in [5]) in order to be able to verify security properties of CPSs either before system deployment or, when static analysis is not feasible, at runtime to promptly detect undesired behaviours. In other words, we aim at providing an essential stepping stone for formal and automated analysis techniques for checking the security of CPSs (rather than for providing defence techniques, i.e., *mitigation* [45]).

Our contribution is fourfold. The *first contribution* is the definition of a *hybrid process calculus*, called CCPSA, to formally specify both CPSs and *physics-based attacks*. In CCPSA, CPSs have two components:

- a *physical component* denoting the *physical plant* (also called environment) of the system, and containing information on state variables, actuators, sensors, evolution law, etc., and
- a *cyber component* that governs access to sensors and actuators, and channel-based communication with other cyber components.

Thus, channels are used for logical interactions between cyber components, whereas sensors and actuators make possible the interaction between cyber and physical components.

CCPSA adopts a *discrete notion of time* [28] and it is equipped with a *labelled transition semantics (LTS)* that allows us to observe both *physical events* (system deadlock and violations of safety conditions) and *cyber events* (channel communications). Based on our LTS, we define two *compositional trace-based system preorders*: a deadlock-sensitive *trace preorder*, \sqsubseteq , and a *timed variant*, $\sqsubseteq_{m..n}$, for $m \in \mathbb{N}^+$ and $n \in \mathbb{N}^+ \cup \{\infty\}$, which takes into account discrepancies of execution traces within the discrete time interval $m..n$. Intuitively, given two CPSs Sys_1 and Sys_2 , we write $Sys_1 \sqsubseteq_{m..n} Sys_2$ if Sys_2 simulates the execution traces of Sys_1 , except for the time interval $m..n$; if $n = \infty$ then the simulation only holds for the first $m - 1$ time slots.

As a *second contribution*, we formalise a *threat model* that specifies *man-in-the-middle (MITM) attacks* that can manipulate *sensor readings* or *control commands* in order to drive a CPS into an undesired state [54].² Without loss of generality, MITM attacks targeting physical devices (sensors or actuators) can be assimilated to *physical attacks*, i.e., those attacks that directly compromise physical devices (e.g., electromagnetic attacks). As depicted in Figure 1, our attacks may affect directly the sensor measurements or the controller commands:

- *Attacks on sensors* consist of reading and eventually replacing y_k (the sensor measurements) with y_k^a .
- *Attacks on actuators* consist of reading, dropping and eventually replacing the controller commands u_k with u_k^a , affecting directly the actions the actuators may execute.

We group attacks into classes. A *class of attacks* takes into account both the potential malicious activities \mathcal{I} on physical devices and the *timing parameters* m and n of the attack: begin and end of the attack. We represent a class C as a total function $C \in [\mathcal{I} \rightarrow \mathcal{P}(m..n)]$. Intuitively, for $\iota \in \mathcal{I}$, $C(\iota) \subseteq m..n$ denotes the set of time instants when an attack of class C may tamper with the device ι .

²Note that we focus on attackers who have already entered the CPS, and we do not consider how they gained access to the system (e.g., by attacking an Internet-accessible controller or one of the communication protocols as a Dolev-Yao-style attacker [17] would do).

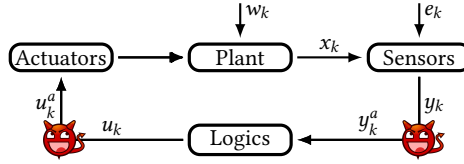


Fig. 1. MITM attacks to sensor readings and control commands

In order to make security assessments on our CPSs, we adopt a well-known approach called *Generalized Non Deducibility on Composition (GNDC)* [18]. Thus, in our calculus CCPSA, we say that a CPS *System* tolerates an attack *A* if

$$\text{Sys} \parallel A \sqsubseteq \text{Sys}.$$

In this case, the presence of the attack *A*, does not change the (physical and logical) observable behaviour of the system *System*, and the attack can be considered harmless.

On the other hand, we say that a CPS *System* is *vulnerable* to an attack *A* of class $C \in [I \rightarrow \mathcal{P}(m..n)]$ if there is a time interval $m'..n'$ in which the attack becomes observable (obviously, $m' \geq m$). Formally, we write:

$$\text{Sys} \parallel A \not\sqsubseteq_{m'..n'} \text{Sys}.$$

We provide sufficient criteria to prove attack tolerance/vulnerability to attacks of an arbitrary class *C*. We define a notion of *most powerful attack* of a given class *C*, $\text{Top}(C)$, and prove that if a CPS tolerates $\text{Top}(C)$ then it tolerates all attacks *A* of class *C* (and of any weaker class³). Similarly, if a CPS is vulnerable to $\text{Top}(C)$, in the time interval $m'..n'$, then no attacks of class *C* (or weaker) can affect the system out of that time interval. This is very useful when checking for attack tolerance/vulnerability with respect to all attacks of a given class *C*.

As a *third contribution*, we formalise how to estimate the *impact of a successful attack* on a CPS. As expected, *risk assessment* in industrial CPSs is a crucial phase preceding any defence strategy implementation [1]. The objective of this phase is to prioritise among vulnerabilities; this is done based on the likelihood that vulnerabilities are exploited, and the impact on the system under attack if exploitation occurs. In this manner, the resources can then be focused on preventing the most critical vulnerabilities [44]. We provide a *metric* to estimate the *maximum perturbation* introduced in the system under attack with respect to its genuine behaviour, according to its evolution law and the uncertainty of the model. Then, we prove that the impact of the most powerful attack $\text{Top}(C)$ represents an upper bound for the impact of any attack *A* of class *C* (or weaker).

Finally, as a *fourth contribution*, we formalise a *running example* in Uppaal SMC [16], the statistical extension of the Uppaal model checker [7] supporting the analysis of systems expressed as composition of *timed and/or probabilistic automata*. Our goal is to test Uppaal SMC as an automatic tool for the *static security analysis* of a simple but significant CPS exposed to a number of different physics-based attacks with different impacts on the system under attack. Here, we wish to remark that while we have kept our running example simple, it is actually non-trivial and designed to describe a wide number of attacks, as will become clear below.

This paper extends and supersedes a preliminary conference version that appeared in [40]. All the results presented in the current paper have been formally proven, although, due to lack of space, proofs of minor statements can be found in the associated technical report [39]. The Uppaal SMC models of our system and the attacks that we have found are available at the repository https://bitbucket.org/AndreiMunteanu/cps_smc/src/.

³Intuitively, attacks of classes weaker than *C* can do less with respect to attacks of class *C*.

1.4 Organisation

In Section 2, we give syntax and semantics of CCPSA. In Section 3, we provide our running example and its formalisation in Uppaal SMC. In Section 4, we first define our threat model for physics-based attacks, then we use Uppaal SMC to carry out a security analysis of our running example when exposed to three different attacks, and, finally, we provide sufficient criteria for attack tolerance/vulnerability. In Section 5, we estimate the impact of attacks on CPSs and prove that the most powerful attack of a given class has the maximum impact with respect to all attacks of the same class (or of a weaker one). In Section 6, we draw conclusions and discuss related and future work.

2 THE CALCULUS

In this section, we introduce our *Calculus of Cyber-Physical Systems and Attacks*, CCPSA, which extends the *Calculus of Cyber-Physical Systems*, defined in our companion papers [38, 42], with specific features to formalise and study attacks to physical devices.

Let us start with some preliminary notation.

2.1 Syntax of CCPSA

NOTATION 1. We use x, x_k for state variables (associated to physical states of systems), c, d for communication channels, a, a_k for actuator devices, and s, s_k or sensors devices.

Actuator names are metavariables for actuator devices like valve, light, etc. Similarly, sensor names are metavariables for sensor devices, e.g., a sensor thermometer that measures a state variable called temperature, with a given precision.

Values, ranged over by v, v', w , are built from basic values, such as Booleans, integers and real numbers; they also include names.

Given a generic set of names N , we write \mathbb{R}^N to denote the set of functions assigning a real value to each name in N . For $\xi \in \mathbb{R}^N$, $n \in N$ and $v \in \mathbb{R}$, we write $\xi[n \mapsto v]$ to denote the function $\psi \in \mathbb{R}^N$ such that $\psi(m) = \xi(m)$, for any $m \neq n$, and $\psi(n) = v$. Given two generic functions ξ_1 and ξ_2 with disjoint domains N_1 and N_2 , respectively, we denote with $\xi_1 \cup \xi_2$ the function such that $(\xi_1 \cup \xi_2)(n) = \xi_1(n)$, if $n \in N_1$, and $(\xi_1 \cup \xi_2)(n) = \xi_2(n)$, if $n \in N_2$.

In general, a cyber-physical system consists of: (i) a *physical component* (defining state variables, physical devices, physical evolution, etc.) and (ii) a *cyber (or logical) component* that interacts with the physical devices (sensors and actuators) and communicates with other cyber components of the same or of other CPSs.

Physical components in CCPSA are given by two sub-components: (i) the *physical state*, which is supposed to change at runtime, and (ii) the *physical environment*, which contains static information.⁴

DEFINITION 1 (PHYSICAL STATE). Let X be a set of state variables, S be a set of sensors, and \mathcal{A} be a set of actuators. A physical state S is a triple $\langle \xi_x, \xi_s, \xi_a \rangle$, where:

- $\xi_x \in \mathbb{R}^X$ is the state function,
- $\xi_s \in \mathbb{R}^S$ is the sensor function,
- $\xi_a \in \mathbb{R}^{\mathcal{A}}$ is the actuator function.

All functions defining a physical state are total.

The *state function* ξ_x returns the current value associated to each variable in X , the *sensor function* ξ_s returns the current value associated to each sensor in S and the *actuator function* ξ_a returns the current value associated to each actuator in \mathcal{A} .

⁴Actually, this information is periodically updated (say, every six months) to take into account possible drifts of the system.

246 DEFINITION 2 (PHYSICAL ENVIRONMENT). Let \mathcal{X} be a set of state variables, \mathcal{S} be a set of sensors, and
 247 \mathcal{A} be a set of actuators. A physical environment E is a 6-tuple $\langle \text{evol}, \text{meas}, \text{inv}, \text{safe}, \xi_w, \xi_e \rangle$, where:

- 248 • $\text{evol} : \mathbb{R}^{\mathcal{X}} \times \mathbb{R}^{\mathcal{A}} \times \mathbb{R}^{\mathcal{X}} \rightarrow 2^{\mathbb{R}^{\mathcal{X}}}$ is the evolution map,
- 249 • $\text{meas} : \mathbb{R}^{\mathcal{X}} \times \mathbb{R}^{\mathcal{S}} \rightarrow 2^{\mathbb{R}^{\mathcal{S}}}$ is the measurement map,
- 250 • $\text{inv} \in 2^{\mathbb{R}^{\mathcal{X}}}$ is the invariant set,
- 251 • $\text{safe} \in 2^{\mathbb{R}^{\mathcal{X}}}$ is the safety set,
- 252 • $\xi_w \in \mathbb{R}^{\mathcal{X}}$ is the uncertainty function,
- 253 • $\xi_e \in \mathbb{R}^{\mathcal{S}}$ is the sensor-error function.

254 All functions defining a physical environment are total functions.
 255

256 The *evolution map* evol models the *evolution law* of the physical system, where changes made
 257 on actuators may reflect on state variables. Given a state function, an actuator function, and an
 258 uncertainty function, the *evolution map* evol returns the set of next *admissible state functions*. Since
 259 we assume an *uncertainty* in our models, evol does not return a single state function but a set of
 260 possible state functions.

261 The *measurement map* meas returns the set of next *admissible sensor functions* based on the
 262 current state function. Since we assume error-prone sensors, meas does not return a single sensor
 263 function but a set of possible sensor functions.

264 The *invariant set* inv represents the set of state functions that satisfy the invariant of the system.
 265 A CPS that gets into a physical state with a state function that does not satisfy the invariant is in
 266 *deadlock*. Similarly, the *safety set* safe represents the set of state functions that satisfy the safety
 267 conditions of the system. Intuitively, if a CPS gets into an unsafe state, then its functionality may
 268 get compromised.

269 The *uncertainty function* ξ_w returns the uncertainty (or accuracy) associated to each state variable.
 270 Thus, given a state variable $x \in \mathcal{X}$, $\xi_w(x)$ returns the maximum distance between the real value of
 271 x , in an arbitrary moment in time, and its representation in the model. For $\xi_w, \xi'_w \in \mathbb{R}^{\mathcal{X}}$, we will
 272 write $\xi_w \leq \xi'_w$ if $\xi_w(x) \leq \xi'_w(x)$, for any $x \in \mathcal{X}$. The evolution map evol is obviously *monotone* with
 273 respect to uncertainty: if $\xi_w \leq \xi'_w$ then $\text{evol}(\xi_x, \xi_a, \xi_w) \subseteq \text{evol}(\xi_x, \xi_a, \xi'_w)$.

274 Finally, the *sensor-error function* ξ_e returns the maximum error associated to each sensor in \mathcal{S} .

275 Let us now define formally the cyber component of a CPS in CCPSA. Our (logical) processes build
 276 on Hennessy and Regan's *Timed Process Language TPL* [28], basically, CCS enriched with a discrete
 277 notion of time. We extend TPL with two main ingredients:

- 278 • two constructs to read values detected at sensors and write values on actuators, respectively;
- 279 • special constructs to represent malicious activities on physical devices.

280 The remaining constructs are the same as those of TPL.
 281

282 DEFINITION 3 (PROCESSES). Processes are defined as follows:
 283

$$\begin{aligned}
 284 \quad P, Q & ::= \text{nil} \mid \text{tick}.P \mid P \parallel Q \mid \pi.P \mid \phi.P \mid [\mu.P]Q \mid \text{if } (b) \{P\} \text{ else } \{Q\} \mid P \setminus c \mid H\langle \tilde{w} \rangle \\
 285 \quad \pi & ::= \text{rcv } c(x) \mid \text{snd } c\langle v \rangle \\
 286 \quad \phi & ::= \text{read } s(x) \mid \text{write } a\langle v \rangle \\
 287 \quad \mu & ::= \text{sniff } s(x) \mid \text{drop } a(x) \mid \text{forge } p\langle v \rangle.
 \end{aligned}$$

288 We write nil for the *terminated process*. The process $\text{tick}.P$ sleeps for one time unit and then
 289 continues as P . We write $P \parallel Q$ to denote the *parallel composition* of concurrent *threads* P and Q .
 290 The process $\pi.P$ denotes *channel transmission*. The construct $\phi.P$ denotes activities on *physical*
 291 *devices*, i.e., *sensor reading* and *actuator writing*. The process $[\mu.P]Q$ denotes MITM malicious
 292 activities under timeout targeting physical devices (sensors and actuators). More precisely, we
 293
 294

support *sensor sniffing*, *drop of actuator commands*, and *integrity attacks on data coming from sensors and addressed to actuators*. Thus, for instance, $\lfloor \text{drop } a(x).P \rfloor Q$ drops a command on the actuator a supplied by the controller in the current time slot; otherwise, if there are no commands on a , it moves to the next time slot and evolves into Q .

The process $P \setminus c$ is the channel restriction operator of CCS. We sometimes write $P \setminus \{c_1, c_2, \dots, c_n\}$ to mean $P \setminus c_1 \setminus c_2 \dots \setminus c_n$. The process $\text{if } (b) \{P\} \text{ else } \{Q\}$ is the standard conditional, where b is a decidable guard. In processes of the form $\text{tick}.Q$ and $\lfloor \mu.P \rfloor Q$, the occurrence of Q is said to be *time-guarded*. The process $H(\tilde{w})$ denotes (guarded) recursion.

We assume a set of *process identifiers* ranged over by H, H_1, H_2 . We write $H(w_1, \dots, w_k)$ to denote a recursive process H defined via an equation $H(x_1, \dots, x_k) = P$, where (i) the tuple x_1, \dots, x_k contains all the variables that appear free in P , and (ii) P contains only guarded occurrences of the process identifiers, such as H itself. We say that recursion is *time-guarded* if P contains only time-guarded occurrences of the process identifiers. Unless explicitly stated our recursive processes are always time-guarded.

In the constructs $\text{rcv } c(x).P$, $\text{read } s(x).P$, $\lfloor \text{sniff } s(x).P \rfloor Q$ and $\lfloor \text{drop } a(x).P \rfloor Q$ the variable x is said to be *bound*. This gives rise to the standard notions of *free/bound (process) variables* and α -*conversion*. A term is *closed* if it does not contain free variables, and we assume to always work with closed processes: the absence of free variables is preserved at run-time. As further notation, we write $T\{v/x\}$ for the substitution of all occurrences of the free variable x in T with the value v .

Everything is in place to provide the definition of cyber-physical systems expressed in CCPSA.

DEFINITION 4 (CYBER-PHYSICAL SYSTEM). *Fixed a set of state variables X , a set of sensors \mathcal{S} , and a set of actuators \mathcal{A} , a cyber-physical system in CCPSA is given by two main components:*

- a physical component consisting of
 - a physical environment E defined on X , \mathcal{S} , and \mathcal{A} , and
 - a physical state S recording the current values associated to the state variables in X , the sensors in \mathcal{S} , and the actuators in \mathcal{A} ;
- a cyber component P that interacts with the sensors in \mathcal{S} and the actuators \mathcal{A} , and can communicate, via channels, with other cyber components of the same or of other CPSs.

We write $E; S \bowtie P$ to denote the resulting CPS, and use M and N to range over CPSs. Sometimes, when the physical environment E is clearly identified, we write $S \bowtie P$ instead of $E; S \bowtie P$. CPSs of the form $S \bowtie P$ are called *environment-free CPSs*.

The syntax of our CPSs is slightly too permissive as a process might use sensors and/or actuators that are not defined in the physical state. To rule out ill-formed CPSs, we use the following definition.

DEFINITION 5 (WELL-FORMEDNESS). *Let $E = \langle \text{evol}, \text{meas}, \text{inv}, \text{safe}, \xi_w, \xi_c \rangle$ be a physical environment, let $S = \langle \xi_x, \xi_s, \xi_a \rangle$ be a physical state defined on a set of physical variables X , a set of sensors \mathcal{S} , and a set of actuators \mathcal{A} , and let P be a process. The CPS $E; S \bowtie P$ is said to be well-formed if: (i) any sensor mentioned in P is in the domain of the function ξ_s ; (ii) any actuator mentioned in P is in the domain of the function ξ_a .*

In the rest of the paper, we will always work with well-formed CPSs and use the following abbreviations.

NOTATION 2. *We write $\mu.P$ for the process defined via the equation $Q = \lfloor \mu.P \rfloor Q$, where Q does not occur in P . Further, we write*

- $\lfloor \mu \rfloor Q$ as an abbreviation for $\lfloor \mu.\text{nil} \rfloor Q$,
- $\lfloor \mu.P \rfloor$ as an abbreviation for $\lfloor \mu.P \rfloor \text{nil}$,
- $\text{snd } c$ and $\text{rcv } c$, when channel c is used for pure synchronisation,

Table 1. LTS for processes

$\text{(Inpp)} \frac{-}{\text{rcv } c(x).P \xrightarrow{cv} P\{v/x\}}$	$\text{(Outp)} \frac{-}{\text{snd } c\langle v \rangle.P \xrightarrow{\bar{c}v} P}$
$\text{(Com)} \frac{P \xrightarrow{\bar{c}v} P' \quad Q \xrightarrow{cv} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$	$\text{(Par)} \frac{P \xrightarrow{\lambda} P' \quad \lambda \neq \text{tick}}{P \parallel Q \xrightarrow{\lambda} P' \parallel Q}$
$\text{(Read)} \frac{-}{\text{read } s(x).P \xrightarrow{s?v} P\{v/x\}}$	$\text{(Write)} \frac{-}{\text{write } a\langle v \rangle.P \xrightarrow{a!v} P}$
$\text{(\$Sniff\$)} \frac{-}{[\text{sniff } s(x).P]Q \xrightarrow{\$s?v} P\{v/x\}}$	$\text{(\$Drop\$)} \frac{-}{[\text{drop } a(x).P]Q \xrightarrow{\$a?v} P\{v/x\}}$
$\text{(\$Forge\$)} \frac{p \in \{s, a\}}{[\text{forge } p\langle v \rangle.P]Q \xrightarrow{\$p!v} P}$	$\text{(\$ActDrop\$)} \frac{P \xrightarrow{a!v} P' \quad Q \xrightarrow{\$a?v} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$
$\text{(Res)} \frac{P \xrightarrow{\lambda} P' \quad \lambda \notin \{cv, \bar{c}v\}}{P \setminus c \xrightarrow{\lambda} P' \setminus c}$	$\text{(\$SensIntegr\$)} \frac{P \xrightarrow{\$s!v} P' \quad Q \xrightarrow{s?v} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$
$\text{(Then)} \frac{\llbracket b \rrbracket = \text{true} \quad P \xrightarrow{\lambda} P'}{\text{if } (b) \{P\} \text{ else } \{Q\} \xrightarrow{\lambda} P'}$	$\text{(Rec)} \frac{P\{\tilde{w}/\tilde{x}\} \xrightarrow{\lambda} Q \quad H(\tilde{x}) = P}{H\langle \tilde{w} \rangle \xrightarrow{\lambda} Q}$
$\text{(TimeNil)} \frac{-}{\text{nil} \xrightarrow{\text{tick}} \text{nil}}$	$\text{(Else)} \frac{\llbracket b \rrbracket = \text{false} \quad Q \xrightarrow{\lambda} Q'}{\text{if } (b) \{P\} \text{ else } \{Q\} \xrightarrow{\lambda} Q'}$
$\text{(Timeout)} \frac{-}{[\mu.P]Q \xrightarrow{\text{tick}} Q}$	$\text{(Sleep)} \frac{-}{\text{tick}.P \xrightarrow{\text{tick}} P}$
$\text{(TimePar)} \frac{P \xrightarrow{\text{tick}} P' \quad Q \xrightarrow{\text{tick}} Q'}{P \parallel Q \xrightarrow{\text{tick}} P' \parallel Q'}$	

- $\text{tick}^k.P$ as a shorthand for $\text{tick} \dots \text{tick}.P$, where the prefix tick appears $k \geq 0$ consecutive times.

Finally, let $M = E; S \bowtie P$, we write $M \parallel Q$ for $E; S \bowtie (P \parallel Q)$, and $M \setminus c$ for $E; S \bowtie (P \setminus c)$.

2.2 Labelled transition semantics

In this subsection, we provide the dynamics of CCPSA in terms of a *labelled transition system (LTS)* in the SOS style of Plotkin. First, we give in Table 1 an LTS for logical processes, then in Table 2 we lift transition rules from processes to environment-free CPSs.

In Table 1, the meta-variable λ ranges over labels in the set $\{\text{tick}, \tau, \bar{c}v, cv, a!v, s?v, \$p!v, \$p?v\}$. Rules (Outp), (Inpp) and (Com) serve to model channel communication, on some channel c . Rules (Read) and (Write) denote sensor reading and actuator writing, respectively. The following three rules model three different MITM malicious activities: sensor sniffing, dropping of actuator commands, and integrity attacks on data coming from sensors or addressed to actuators. In particular, rule ($\$ActDrop\$$) models a *DoS attack to the actuator a*, where the update request of the controller is dropped by the attacker and it never reaches the actuator, whereas rule ($\$SensIntegr\$$) models an *integrity attack on sensor s*, as the controller of s is supplied with a fake value v forged by the

Table 2. LTS for CPSs $S \bowtie P$ parametric on an environment $E = \langle evol, meas, inv, safe, \xi_w, \xi_e \rangle$

$$\begin{array}{c}
393 \\
394 \\
395 \\
396 \\
397 \\
398 \\
399 \\
400 \\
401 \\
402 \\
403 \\
404 \\
405 \\
406 \\
407 \\
408 \\
409 \\
410 \\
411 \\
412 \\
413 \\
414 \\
415 \\
416 \\
417 \\
418 \\
419 \\
420 \\
421 \\
422 \\
423 \\
424 \\
425 \\
426 \\
427 \\
428 \\
429 \\
430 \\
431 \\
432 \\
433 \\
434 \\
435 \\
436 \\
437 \\
438 \\
439 \\
440 \\
441
\end{array}$$

$$\begin{array}{l}
\text{(Out)} \frac{S = \langle \xi_x, \xi_s, \xi_a \rangle \quad P \xrightarrow{\bar{c}v} P' \quad \xi_x \in inv}{S \bowtie P \xrightarrow{\bar{c}v} S \bowtie P'} \qquad \text{(Inp)} \frac{S = \langle \xi_x, \xi_s, \xi_a \rangle \quad P \xrightarrow{cv} P' \quad \xi_x \in inv}{S \bowtie P \xrightarrow{cv} S \bowtie P'} \\
\text{(SensRead)} \frac{P \xrightarrow{s?v} P' \quad \xi_s(s) = v \quad P \not\xrightarrow{\xi_s!v} \quad \xi_x \in inv}{\langle \xi_x, \xi_s, \xi_a \rangle \bowtie P \xrightarrow{\tau} \langle \xi_x, \xi_s, \xi_a \rangle \bowtie P'} \\
\text{(\textit{\textless}SensSniff\textit{\textless})} \frac{P \xrightarrow{\xi_s?v} P' \quad \xi_s(s) = v \quad \xi_x \in inv}{\langle \xi_x, \xi_s, \xi_a \rangle \bowtie P \xrightarrow{\tau} \langle \xi_x, \xi_s, \xi_a \rangle \bowtie P'} \\
\text{(ActWrite)} \frac{P \xrightarrow{a!v} P' \quad \xi'_a = \xi_a[a \mapsto v] \quad P \not\xrightarrow{\xi_a?v} \quad \xi_x \in inv}{\langle \xi_x, \xi_s, \xi_a \rangle \bowtie P \xrightarrow{\tau} \langle \xi_x, \xi_s, \xi'_a \rangle \bowtie P'} \\
\text{(\textit{\textless}ActIntegr\textit{\textless})} \frac{P \xrightarrow{\xi_a!v} P' \quad \xi'_a = \xi_a[a \mapsto v] \quad \xi_x \in inv}{\langle \xi_x, \xi_s, \xi_a \rangle \bowtie P \xrightarrow{\tau} \langle \xi_x, \xi_s, \xi'_a \rangle \bowtie P'} \\
\text{(Tau)} \frac{P \xrightarrow{\tau} P' \quad \xi_x \in inv}{\langle \xi_x, \xi_s, \xi_a \rangle \bowtie P \xrightarrow{\tau} \langle \xi_x, \xi_s, \xi_a \rangle \bowtie P'} \qquad \text{(Deadlock)} \frac{S = \langle \xi_x, \xi_s, \xi_a \rangle \quad \xi_x \notin inv}{S \bowtie P \xrightarrow{\text{deadlock}} S \bowtie P} \\
\text{(Time)} \frac{P \xrightarrow{\text{tick}} P' \quad S = \langle \xi_x, \xi_s, \xi_a \rangle \quad S' \in \text{next}(E; S) \quad \xi_x \in inv}{S \bowtie P \xrightarrow{\text{tick}} S' \bowtie P'} \\
\text{(Safety)} \frac{S = \langle \xi_x, \xi_s, \xi_a \rangle \quad \xi_x \notin \text{safe} \quad \xi_x \in inv}{S \bowtie P \xrightarrow{\text{unsafe}} S \bowtie P}
\end{array}$$

attack. Rule (Par) propagates untimed actions over parallel components. Rules (Res), (Rec), (Then) and (Else) are standard. The following four rules (TimeNil), (Sleep), (TimeOut) and (TimePar) model the passage of time. For simplicity, we omit the symmetric counterparts of the rules (Com), (ξ ActDrop ξ), (ξ SensIntegr ξ), and (Par).

In Table 2, we lift the transition rules from processes to environment-free CPSs of the form $S \bowtie P$ for $S = \langle \xi_x, \xi_s, \xi_a \rangle$. The transition rules are parametric on a physical environment E . Except for rule (Deadlock), all rules have a common premise $\xi_x \in inv$: a system can evolve only if the invariant is satisfied by the current physical state. Here, actions, ranged over by α , are in the set $\{\tau, \bar{c}v, cv, \text{tick}, \text{deadlock}, \text{unsafe}\}$. These actions denote: internal activities (τ); channel transmission ($\bar{c}v$ and cv); the passage of time (tick); and two specific physical events: system deadlock (deadlock) and the violation of the safety conditions (unsafe). Rules (Out) and (Inp) model transmission and reception, with an external system, on a channel c . Rule (SensRead) models the reading of the current data detected at a sensor s ; here, the presence of a malicious action $\xi_s!w$ would prevent the reading of the sensor. We already said that rule (ξ SensIntegr ξ) of Table 1 models integrity attacks on a sensor s . However, together with rule (SensRead), it also serves to implicitly model *DoS attacks on a sensor* s , as the controller of s cannot read its correct value if the attacker is currently supplying a fake value for it. Rule (ξ SensSniff ξ) allows the attacker to read the confidential value detected at a sensor s . Rule (ActWrite) models the writing of a value v on an actuator a ; here, the presence of an attack capable of performing a drop action $\xi_a?v$ prevents the access to the actuator by the controller. Rule (ξ ActIntegr ξ) models a *MITM integrity attack to an actuator* a , as the actuator is provided with

a value forged by the attack. Rule (Tau) lifts non-observable actions from processes to systems. This includes communications channels and attacks' accesses to physical devices. A similar lifting occurs in rule (Time) for timed actions, where $\text{next}(E; S)$ returns the set of possible physical states for the next time slot. Formally, for $E = \langle \text{evol}, \text{meas}, \text{inv}, \text{safe}, \xi_w, \xi_e \rangle$ and $S = \langle \xi_x, \xi_s, \xi_a \rangle$, we define:

$$\text{next}(E; S) \stackrel{\text{def}}{=} \{ \langle \xi'_x, \xi'_s, \xi'_a \rangle : \xi'_x \in \text{evol}(\xi_x, \xi_a, \xi_w) \wedge \xi'_s \in \text{meas}(\xi'_x, \xi_e) \wedge \xi'_a = \xi_a \}.$$

Thus, by an application of rule (Time) a CPS moves to the next physical state, in the next time slot. Rule (Deadlock) is introduced to signal the violation of the invariant. When the invariant is violated, a system deadlock occurs and then, in CCPSA, the system emits a special action `deadlock`, forever. Similarly, rule (Safety) is introduced to detect the violation of safety conditions. In this case, the system may emit a special action `unsafe` and then continue its evolution.

Summarising, in the LTS of Table 2 we define transitions rules of the form $S \bowtie P \xrightarrow{\alpha} S' \bowtie P'$, parametric on some physical environment E . As physical environments do not change at runtime, $S \bowtie P \xrightarrow{\alpha} S' \bowtie P'$ entails $E; S \bowtie P \xrightarrow{\alpha} E; S' \bowtie P'$, thus providing the LTS for all CPSs in CCPSA.

REMARK 1. *Note that our operational semantics ensures that malicious actions of the form $\ell s!v$ (integrity/DoS attack on sensor s) or $\ell a?v$ (DoS attack on actuator a) have a pre-emptive power. These attacks can always prevent the regular access to a physical device by its controller.*

2.3 Behavioural semantics

Having defined the actions that can be performed by a CPS of the form $E; S \bowtie P$, we can easily concatenate these actions to define the possible *execution traces* of the system. Formally, given a trace $t = \alpha_1 \dots \alpha_n$, we will write \xrightarrow{t} as an abbreviation for $\xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n}$, and we will use the function $\#\text{tick}(t)$ to get the number of occurrences of the action `tick` in t .

The notion of trace allows us to provide a formal definition of system soundness: a CPS is said to be *sound* if it never deadlocks and never violates the safety conditions.

DEFINITION 6 (SYSTEM SOUNDNESS). *Let M be a well-formed CPS. We say that M is sound if whenever $M \xrightarrow{t} M'$, for some t , the actions `deadlock` and `unsafe` never occur in t .*

In our security analysis, we will always focus on sound CPSs.

We recall that the *observable activities* in CCPSA are: time passing, system deadlock, violation of safety conditions, and channel communication. Having defined a labelled transition semantics, we are ready to formalise our behavioural semantics, based on execution traces.

We adopt a standard notation for weak transitions: we write \Rightarrow for $(\xrightarrow{\tau})^*$, whereas $\xRightarrow{\alpha}$ means $\Rightarrow \xrightarrow{\alpha} \Rightarrow$, and finally $\xRightarrow{\hat{\alpha}}$ denotes \Rightarrow if $\alpha = \tau$ and $\xRightarrow{\alpha}$ otherwise. Given a trace $t = \alpha_1 \dots \alpha_n$, we write $\xRightarrow{\hat{t}}$ as an abbreviation for $\xRightarrow{\hat{\alpha}_1} \dots \xRightarrow{\hat{\alpha}_n}$.

DEFINITION 7 (TRACE PREORDER). *We write $M \sqsubseteq N$ if whenever $M \xrightarrow{t} M'$, for some t , there is N' such that $N \xRightarrow{\hat{t}} N'$.*

REMARK 2. *Unlike other process calculi, in CCPSA our trace preorder is able to observe (physical) deadlock due to the presence of the rule (Deadlock) and the special action `deadlock`: whenever $M \sqsubseteq N$ then M eventually deadlocks if and only if N eventually deadlocks (see Lemma 1 in the appendix).*

Our trace preorder can be used for *compositional reasoning* in those contexts that don't interfere on physical devices (sensors and actuators) while they may interfere on logical components (via channel communication). In particular, trace preorder is preserved by parallel composition of *physically-disjoint* CPSs, by parallel composition of *pure-logical* processes, and by channel restriction.

Intuitively, two CPSs are physically-disjoint if they have different plants but they may share logical channels for communication purposes. More precisely, physically-disjoint CPSs have disjoint state variables and disjoint physical devices (sensors and actuators). As we consider only well-formed CPSs (Definition 5), this ensures that a CPS cannot physically interfere with a parallel CPS by acting on its physical devices.

Formally, let $S_i = \langle \xi_x^i, \xi_s^i, \xi_a^i \rangle$ and $E_i = \langle evol^i, meas^i, inv^i, safe^i, \xi_w^i, \xi_e^i \rangle$ be physical states and physical environments, respectively, associated to sets of state variables \mathcal{X}_i , sets of sensors \mathcal{S}_i , and sets of actuators \mathcal{A}_i , for $i \in \{1, 2\}$. For $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$, $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ and $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$, we define:

- the *disjoint union* of the physical states S_1 and S_2 , written $S_1 \uplus S_2$, to be the physical state $\langle \xi_x, \xi_s, \xi_a \rangle$ such that: $\xi_x = \xi_x^1 \cup \xi_x^2$, $\xi_s = \xi_s^1 \cup \xi_s^2$, and $\xi_a = \xi_a^1 \cup \xi_a^2$;
- the *disjoint union* of the physical environments E_1 and E_2 , written $E_1 \uplus E_2$, to be the physical environment $\langle evol, meas, inv, safe, \xi_w, \xi_e \rangle$ such that:
 - (1) $evol = evol^1 \cup evol^2$
 - (2) $meas = meas^1 \cup meas^2$
 - (3) $S_1 \uplus S_2 \in inv$ iff $S_1 \in inv^1$ and $S_2 \in inv^2$
 - (4) $S_1 \uplus S_2 \in safe$ iff $S_1 \in safe^1$ and $S_2 \in safe^2$
 - (5) $\xi_w = \xi_w^1 \cup \xi_w^2$
 - (6) $\xi_e = \xi_e^1 \cup \xi_e^2$.

DEFINITION 8 (PHYSICALLY-DISJOINT CPSs). Let $M_i = E_i; S_i \bowtie P_i$, for $i \in \{1, 2\}$. We say that M_1 and M_2 are physically-disjoint if S_1 and S_2 have disjoint sets of state variables, sensors and actuators. In this case, we write $M_1 \uplus M_2$ to denote the CPS defined as $(E_1 \uplus E_2); (S_1 \uplus S_2) \bowtie (P_1 \parallel P_2)$.

A *pure-logical process* is a process that may interfere on communication channels but it never interferes on physical devices as it never accesses sensors and/or actuators. Basically, a pure-logical process is a TPL process [28]. Thus, in a system $M \parallel Q$, where M is an arbitrary CPS, a pure-logical process Q cannot interfere with the physical evolution of M . A process Q can, however, definitely interact with M via communication channels, and hence affect its observable behaviour.

DEFINITION 9 (PURE-LOGICAL PROCESSES). A process P is called pure-logical if it never acts on sensors and/or actuators.

Now, we can finally state the compositionality of our trace preorder \sqsubseteq (the proof can be found in the appendix).

THEOREM 1 (COMPOSITIONALITY OF \sqsubseteq). Let M and N be two arbitrary CPSs in CCPSA.

- (1) $M \sqsubseteq N$ implies $M \uplus O \sqsubseteq N \uplus O$, for any physically-disjoint CPS O ;
- (2) $M \sqsubseteq N$ implies $M \parallel P \sqsubseteq N \parallel P$, for any pure-logical process P ;
- (3) $M \sqsubseteq N$ implies $M \setminus c \sqsubseteq N \setminus c$, for any channel c .

The reader may wonder whether our trace preorder \sqsubseteq is preserved by more permissive contexts. The answer is no. Suppose that in the second item of Theorem 1 we allowed a process P that can also read on sensors. In this case, even if $M \sqsubseteq N$, the parallel process P might read a different value in the two systems at the very same sensor s (due to the sensor error) and transmit these different values on a free channel, breaking the congruence. Activities on actuators may also lead to different behaviours of the compound systems: M and N may have physical components that are not exactly aligned. A similar reasoning applies when composing CPSs with non physically-disjoint ones: noise on physical devices may break the compositionality result.

As we are interested in formalising timing aspects of attacks, such as beginning and duration, we propose a timed variant of \sqsubseteq up to (a possibly infinite) *discrete time interval* $m..n$, with $m \in \mathbb{N}^+$

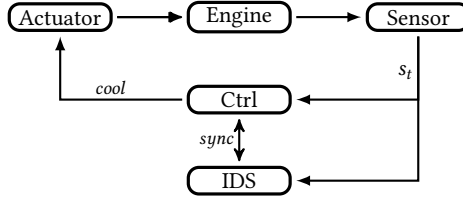


Fig. 2. The main structure of the CPS Sys

and $n \in \mathbb{N}^+ \cup \infty$. Intuitively, we write $M \sqsubseteq_{m..n} N$ if the CPS N simulates the execution traces of M in all time slots, except for those contained in the discrete time interval $m..n$.

DEFINITION 10 (TRACE PREORDER UP TO A TIME INTERVAL). We write $M \sqsubseteq_{m..n} N$, for $m \in \mathbb{N}^+$ and $n \in \mathbb{N}^+ \cup \{\infty\}$, with $m \leq n$, if the following conditions hold:

- m is the minimum integer for which there is a trace t , with $\#\text{tick}(t)=m-1$, s.t. $M \xrightarrow{t}$ and $N \not\xrightarrow{t}$;
- n is the infimum element of $\mathbb{N}^+ \cup \{\infty\}$, $n \geq m$, such that whenever $M \xrightarrow{t_1} M'$, with $\#\text{tick}(t_1) = n-1$, there is t_2 , with $\#\text{tick}(t_1) = \#\text{tick}(t_2)$, such that $N \xrightarrow{t_2} N'$, for some N' , and $M' \sqsubseteq N'$.

In Definition 10, the first item says that N can simulate the traces of M for at most $m-1$ time slots; whereas the second item says two things: (i) in time interval $m..n$ the simulation does not hold; (ii) starting from the time slot $n+1$ the CPS N can simulate again the traces of M . Note that $\inf(\emptyset) = \infty$. Thus, if $M \sqsubseteq_{m..\infty} N$, then N simulates M only in the first $m-1$ time slots.

THEOREM 2 (COMPOSITIONALITY OF $\sqsubseteq_{m..n}$). Let M and N be two arbitrary CPSs in CCPSA.

- (1) $M \sqsubseteq_{m..n} N$ implies that for any physically-disjoint CPS there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M \uplus O \sqsubseteq_{m'..n'} N \uplus O$;
- (2) $M \sqsubseteq_{m..n} N$ implies that for any pure-logical process P there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M \parallel P \sqsubseteq_{m'..n'} N \parallel P$;
- (3) $M \sqsubseteq_{m..n} N$ implies that for any channel c there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M \setminus c \sqsubseteq_{m'..n'} N \setminus c$.

The proof can be found in the appendix.

3 A RUNNING EXAMPLE

In this section, we introduce a running example to illustrate how we can precisely represent CPSs and a variety of different physics-based attacks. In practice, we formalise a relatively simple CPS Sys in which the temperature of an *engine* is maintained within a specific range by means of a cooling system. We wish to remark here that while we have kept the example simple, it is actually far from trivial and designed to describe a wide number of attacks. The main structure of the CPS Sys is shown in Figure 2.

3.1 The CPS Sys

The physical state *State* of the engine is characterised by: (i) a state variable *temp* containing the current temperature of the engine, and an integer state variable *stress* keeping track of the level of stress of the mechanical parts of the engine due to high temperatures (exceeding 9.9 degrees); this integer variable ranges from 0, meaning no stress, to 5, for high stress; (ii) a sensor s_t (such as a thermometer or a thermocouple) measuring the temperature of the engine, (iii) an actuator *cool* to turn on/off the cooling system.

The physical environment of the engine, Env , is constituted by: (i) a simple evolution law $evol$ that increases (respectively, decreases) the value of $temp$ by one degree per time unit, when the cooling system is inactive (respectively, active), up to the uncertainty of the system; the variable $stress$ is increased each time the current temperature is above 9.9 degrees, and dropped to 0 otherwise; (ii) a measurement map $meas$ returning the value detected by the sensor s_t , up to the error associated to the sensor; (iii) an invariant set saying that the system gets faulty when the temperature of the engine gets out of the range $[0, 50]$, (iv) a safety set to express that the system moves to an unsafe state when the level of stress reaches the threshold 5, (v) an uncertainty function in which each state variable may evolve with an uncertainty $\delta = 0.4$ degrees, (vi) a sensor-error function saying that the sensor s_t has an accuracy $\epsilon = 0.1$ degrees.

Formally, $State = \langle \xi_x, \xi_s, \xi_a \rangle$ where:

- $\xi_x \in \mathbb{R}^{\{temp, stress\}}$ and $\xi_x(temp) = 0$ and $\xi_x(stress) = 0$;
- $\xi_s \in \mathbb{R}^{\{s_t\}}$ and $\xi_s(s_t) = 0$;
- $\xi_a \in \mathbb{R}^{\{cool\}}$ and $\xi_a(cool) = \text{off}$; for the sake of simplicity, we can assume ξ_a to be a mapping $\{cool\} \rightarrow \{\text{on}, \text{off}\}$ such that $\xi_a(cool) = \text{off}$ if $\xi_a(cool) \geq 0$, and $\xi_a(cool) = \text{on}$ if $\xi_a(cool) < 0$;

and $Env = \langle evol, meas, inv, safe, \xi_w, \xi_e \rangle$ with:

- $evol(\xi_x^i, \xi_a^i, \xi_w)$ is the set of functions $\xi \in \mathbb{R}^{\{temp, stress\}}$ such that:
 - $\xi(temp) = \xi_x^i(temp) + heat(\xi_a^i, cool) + \gamma$, with $\gamma \in [-\delta, +\delta]$ and $heat(\xi_a^i, cool) = -1$ if $\xi_a^i(cool) = \text{on}$ (active cooling), and $heat(\xi_a^i, cool) = +1$ if $\xi_a^i(cool) = \text{off}$ (inactive cooling);
 - $\xi(stress) = \min(5, \xi_x^i(stress)+1)$ if $\xi_x^i(temp) > 9.9$; $\xi(stress) = 0$, otherwise;
- $meas(\xi_x^i, \xi_e) = \{ \xi : \xi(s_t) \in [\xi_x^i(temp) - \epsilon, \xi_x^i(temp) + \epsilon] \}$;
- $inv = \{ \xi_x^i : 0 \leq \xi_x^i(temp) \leq 50 \}$;
- $safe = \{ \xi_x^i : \xi_x^i(stress) < 5 \}$ (we recall that the stress threshold is 5);
- $\xi_w \in \mathbb{R}^{\{temp, stress\}}$, $\xi_w(temp) = 0.4 = \delta$ and $\xi_w(stress) = 0$;
- $\xi_e \in \mathbb{R}^{\{s_t\}}$ and $\xi_e(s_t) = 0.1 = \epsilon$.

For the cyber component of the CPS Sys , we define two parallel processes: $Ctrl$ and IDS . The former models the *controller* activity, consisting in reading the temperature sensor and in governing the cooling system via its actuator, whereas the latter models a simple *intrusion detection system* that attempts to detect and signal *anomalies in the behaviour of the system* [24]. Intuitively, $Ctrl$ senses the temperature of the engine at each time slot. When the *sensed temperature* is above 10 degrees, the controller activates the coolant. The cooling activity is maintained for 5 consecutive time units. After that time, the controller synchronises with the IDS component via a private channel $sync$, and then waits for *instructions*, via a channel ins . The IDS component checks whether the *sensed temperature* is still above 10. If this is the case, it sends an *alarm* of “high temperature”, via a specific channel, and then tells $Ctrl$ to keep cooling for 5 more time units; otherwise, if the temperature is not above 10, the IDS component requires $Ctrl$ to stop the cooling activity.

```

628 Ctrl = read s_t(x).if (x > 10) {Cooling} else {tick.Ctrl}
629 Cooling = write cool(on).tick5.Check
630 Check = snd sync.rcv ins(y).if (y = keep_cooling) {tick5.Check} else {write cool(off).tick.Ctrl}
631 IDS = rcv sync.read s_t(x).if (x > 10) {snd alarm(high_temp).snd ins(keep_cooling).tick.IDS}
632 else {snd ins(stop).tick.IDS} .
633

```

Thus, the whole CPS is defined as:

$$Sys = Env; State \bowtie (Ctrl \parallel IDS) \setminus \{sync, ins\}$$

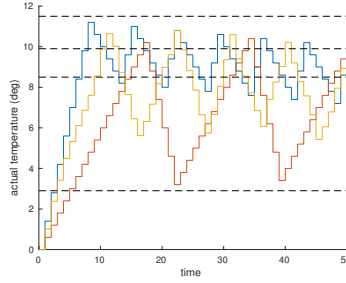


Fig. 3. Three possible evolutions of the CPS Sys

For the sake of simplicity, our *IDS* component is quite basic: for instance, it does not check whether the temperature is too low. However, it is straightforward to replace it with a more sophisticated one, containing more informative tests on sensor values and/or on actuators commands.

Figure 3 shows three possible evolutions in time of the state variable *temp* of *Sys*: (i) the first one (in red), in which the temperature of the engine always grows of $1 - \delta = 0.6$ degrees per time unit, when the cooling is off, and always decrease of $1 + \delta = 1.4$ degrees per time unit, when the cooling is on; (ii) the second one (in blue), in which the temperature always grows of $1 + \delta = 1.4$ degrees per time unit, when the cooling is off, and always decreases of $1 - \delta = 0.6$ degrees per time unit, when the cooling is on; (iii) a third one (in yellow), in which, depending on whether the cooling is off or on, at each time step the temperature grows or decreases of an arbitrary offset lying in the interval $[1 - \delta, 1 + \delta]$.

Our operational semantics allows us to formally prove a number of properties of our running example. For instance, Proposition 1 says that the *Sys* is sound and it never fires the *alarm*.

PROPOSITION 1. *If $Sys \xrightarrow{t}$ for some trace $t = \alpha_1 \dots \alpha_n$, then $\alpha_i \in \{\tau, \text{tick}\}$, for any $i \in \{1, \dots, n\}$.*

Actually, we can be quite precise on the temperature reached by *Sys* before and after the cooling: in each of the 5 rounds of cooling, the temperature will drop of a value lying in the real interval $[1 - \delta, 1 + \delta]$, where δ is the uncertainty.

PROPOSITION 2. *For any execution trace of *Sys*, we have:*

- *when *Sys* turns on the cooling, the value of the state variable *temp* ranges over (9.9, 11.5];*
- *when *Sys* turns off the cooling, the value of the variable *temp* ranges over (2.9, 8.5].*

The proofs of the Propositions 1 and 2 can be found in the associated technical report [39]. In the following section, we will verify the safety properties stated in these two propositions relying on the statistical model checker Uppaal SMC [16].

3.2 A formalisation of *Sys* in Uppaal SMC

In this section, we formalise our running example in Uppaal SMC [16], the *statistical* extension of the Uppaal model checker [7] supporting the analysis of systems expressed as composition of *timed and/or probabilistic automata*. In Uppaal SMC, the user must specify two main statistical parameters α and ϵ , ranging in the interval $[0, 1]$, and representing the probability of *false negatives* and probabilistic *uncertainty*, respectively. Thus, given a CTL property of the system under investigation, the tool returns a probability estimate for that property, lying in a confidence interval $[p - \epsilon, p + \epsilon]$, for some probability $p \in [0, 1]$, with an accuracy $1 - \alpha$. The number of necessary runs to ensure the required accuracy is then computed by the tool relying on the Chernoff-Hoeffding theory [13].

687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735

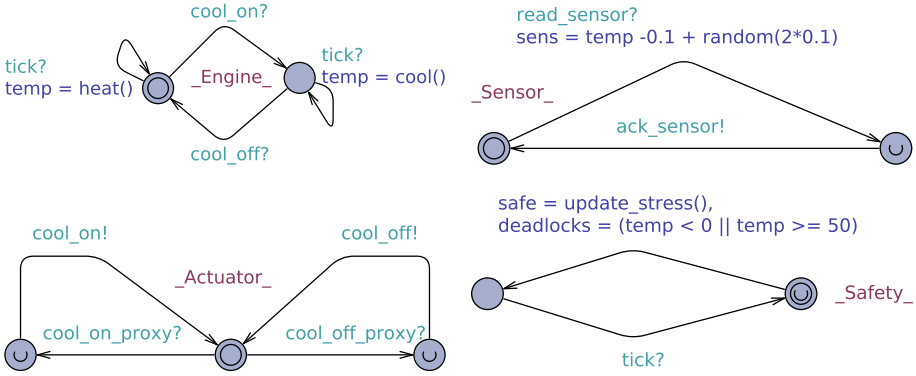


Fig. 4. Uppaal SMC model for the physical component of Sys

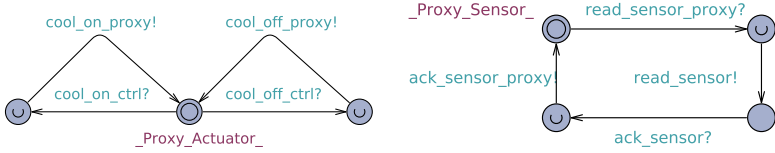


Fig. 5. Uppaal SMC model for the network component of Sys

3.2.1 *Model.* The Uppaal SMC model of our use case Sys is given by three main components represented in terms of *parallel timed automata*: the *physical component*, the *network*, and the *logical component*.

The physical component, whose model is shown in Figure 4, consists of four automata: (i) the *_Engine_* automaton that governs the evolution of the variable *temp* by means of the *heat* and *cool* functions; (ii) the *_Sensor_* automaton that updates the global variable *sens* at each measurement request; (iii) the *_Actuator_* automaton that activates/deactivates the cooling system; (iv) the *_Safety_* automaton that handles the integer variable *stress*, via the *update_stress* function, and the Boolean variables *safe* and *deadlocks*, associated to the safety set *safe* and the invariant set *inv* of Sys, respectively.⁵ We also have a small automaton to model a discrete notion of time (via a synchronisation channel *tick*) as the evolution of state variables is represented via difference equations.

The *network*, whose model is given in Figure 5, consists of *two proxies*: a proxy to relay actuator commands between the actuator device and the controller, a second proxy to relay measurement requests between the sensor device and the logical components (controller and IDS).

The *logical component*, whose model is given in Figure 6, consists of two automata: *_Ctrl_* and *_IDS_* to model the controller and the Intrusion Detection System, respectively; both of them synchronise with their associated proxy copying a fresh value of *sens* into their local variables (*sens_ctrl* and *sens_ids*, respectively). Under proper conditions, the *_IDS_* automaton fires alarms by setting a Boolean variable *alarm*.

3.2.2 *Verification.* We conduct our safety verification using a notebook with the following set-up: (i) 2.8 GHz Intel i7 7700 HQ, with 16 GB memory, and Linux Ubuntu 16.04 operating system;

⁵In Section 6.2, we explain why we need to implement an automaton to check for safety conditions rather than verifying a safety property.

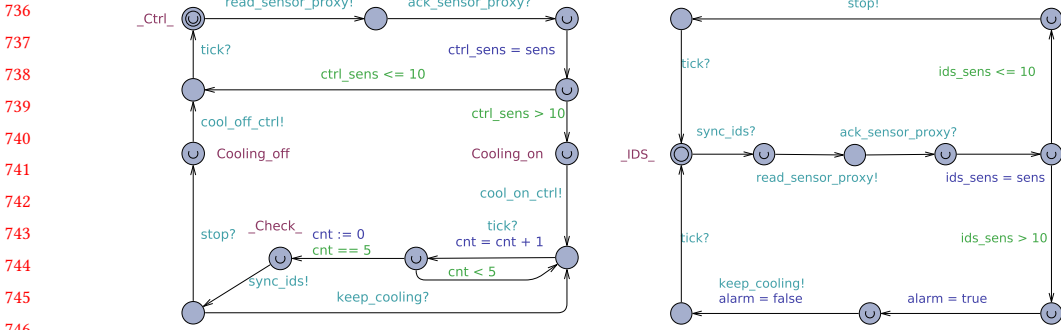


Fig. 6. Uppaal SMC model for the logical component of Sys

(ii) Uppaal SMC model-checker 64-bit, version 4.1.19. The statistical parameters of false negatives (α) and probabilistic uncertainty (ϵ) are both set to 0.01, leading to a confidence level of 99%. As a consequence, having fixed these parameters, for each of our experiments, Uppaal SMC run a number of runs that may vary from a few hundreds to 26492 (*cf.* Chernoff-Hoeffding bounds).

We basically use Uppaal SMC to verify properties expressed in terms of *time bounded CTL* formulae of the form $\square_{[t_1, t_2]} e_{prop}$ and $\diamond_{[0, t_2]} e_{prop}$ ⁶, where t_1 and t_2 are *time instants* according to the discrete representation of time in Uppaal SMC. In practice, we use formulae of the form $\square_{[t_1, t_2]} e_{prop}$ to compute the probability that a property e_{prop} ⁷ holds in all *time slots* of the time interval $t_1..t_2$, whereas with formulae of the form $\diamond_{[0, t_2]} e_{prop}$ we calculate the probability that a property e_{prop} holds in a least one *time slot* of the time interval $0..t_2$.

Thus, instead of proving Proposition 1, we verify, with a 99% accuracy, that in all possible executions that are at most 1000 time slots long, the system Sys results to be sound and alarm free, with probability 0.99. Formally, we verify the following three properties:

- $\square_{[1, 1000]} (\neg \text{deadlocks})$, expressing that the system does not deadlock;
- $\square_{[1, 1000]} (\text{safe})$, expressing that the system does not violate the safety conditions;
- $\square_{[1, 1000]} (\neg \text{alarm})$, expressing that the IDS does not fire any alarm.

Furthermore, instead of Proposition 2, we verify, with the same accuracy and for runs of the same length (up to a short initial transitory phase **lasting 5 time instants**) that if the cooling system is off, then the temperature of the engine lies in the real interval (2.9, 8.5], otherwise it ranges over the interval (9.9, 11.5]. Formally, we verify the following two properties:

- $\square_{[5, 1000]} (\text{Cooling_off} \implies (\text{temp} > 2.9 \wedge \text{temp} \leq 8.5))$
- $\square_{[5, 1000]} (\text{Cooling_on} \implies (\text{temp} > 9.9 \wedge \text{temp} \leq 11.5))$.

The verification of each of the five properties above requires around 15 minutes. The Uppaal SMC models of our system and the attacks discussed in the next section are available at the repository https://bitbucket.org/AndreiMunteanu/cps_smc/src/.

REMARK 3 (ON UPPER-BOUND ANALYSIS). *In our Uppaal SMC model we decided to represent both uncertainty of physical evolution (in the functions heat and cool of `_Engine_`) and measurement noise (in `_Sensor_`) in a probabilistic manner via random extractions. Here, the reader may wonder whether it would have been enough to restrict our SMC analysis by considering only upper bounds on these two quantities. Actually, this is not the case. In fact, suppose to consider only execution traces in which the physical uncertainty is always either 0.4 or -0.4 . Then, the reachable temperatures are always of the*

⁶The 0 in the left-hand side of the time interval is imposed by the syntax of Uppaal SMC.

⁷ e_{prop} is a side-effect free expression over variables (e.g., clock variables, location names and primitive variables) [7].

form $n.k$, for $n, k \in \mathbb{N}$ and k even (e.g., there is no way to reach the maximum admissible temperature of 11.5 degrees).

4 PHYSICS-BASED ATTACKS

In this section, we use CCPSA to formalise a *threat model* of physics-based attacks, i.e., attacks that can manipulate sensor and/or actuator signals in order to drive a *sound* CPS into an undesired state [54]. An attack may have different levels of access to physical devices; for example, it might be able to get read access to the sensors but not write access; or it might get write-only access to the actuators but not read-access. This level of granularity is very important to model precisely how physics-based attacks can affect a CPS [14].

In CCPSA, we have a syntactic way to distinguish malicious processes from honest ones.

DEFINITION 11 (HONEST SYSTEM). *A CPS $E; S \bowtie P$ is honest if P is honest, where P is honest if it does not contain constructs of the form $\lfloor \mu.P_1 \rfloor P_2$.*

We group physics-based attacks in classes that describe both the malicious activities and the timing aspects of the attack. Intuitively, a class of attacks provides information about which physical devices are accessed by the attacks of that class, how they are accessed (read and/or write), when the attack begins and when the attack ends. Thus, let \mathcal{I} be the set of all possible malicious activities on the physical devices of a system, $m \in \mathbb{N}^+$ be the time slot when an attack starts, and $n \in \mathbb{N}^+ \cup \{\infty\}$ be the time slot when the attack ends. We then say that an *attack* A is of class $C \in [\mathcal{I} \rightarrow \mathcal{P}(m..n)]$ if:

- (1) all possible malicious activities of A coincide with those contained in \mathcal{I} ;
- (2) the first of those activities may occur in the m^{th} time slot but not before;
- (3) the last of those activities may occur in the n^{th} time slot but not after;
- (4) for $\iota \in \mathcal{I}$, $C(\iota)$ returns a (possibly empty) set of time slots when A may read/tamper with the device ι (this set is contained in $m..n$);
- (5) C is a total function, i.e., if no attacks of class C can achieve the malicious activity $\iota \in \mathcal{I}$, then $C(\iota) = \emptyset$.

DEFINITION 12 (CLASS OF ATTACKS). *Let $\mathcal{I} = \{\ell p? : p \in \mathcal{S} \cup \mathcal{A}\} \cup \{\ell p! : p \in \mathcal{S} \cup \mathcal{A}\}$ be the set of all possible malicious activities on physical devices. Let $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+ \cup \{\infty\}$, with $m \leq n$. A class of attacks $C \in [\mathcal{I} \rightarrow \mathcal{P}(m..n)]$ is a total function such that for any attack A of class C we have:*

- (i) $C(\iota) = \{k : A \xrightarrow{\iota} \xrightarrow{!v} A' \wedge k = \#tick(t) + 1\}$, for $\iota \in \mathcal{I}$,
- (ii) $m = \inf\{k : k \in C(\iota) \wedge \iota \in \mathcal{I}\}$,
- (iii) $n = \sup\{k : k \in C(\iota) \wedge \iota \in \mathcal{I}\}$.

Along the lines of [18], we can say that an attack A affects a *sound* CPS M if the execution of the compound system $M \parallel A$ differs from that of the original system M , in an observable manner. Basically, a physics-based attack can influence the system under attack in at least two different ways:

- The system $M \parallel A$ might deadlock when M may not; this means that the attack A affects the *availability* of the system. We recall that in the context of CPSs, deadlock is a particular severe physical event.
- The system $M \parallel A$ might have non-genuine execution traces containing observables (violations of safety conditions or communications on channels) that can't be reproduced by M ; here the attack affects the *integrity* of the system behaviour.

DEFINITION 13 (ATTACK TOLERANCE/VULNERABILITY). *Let M be an honest and sound CPS. We say that M is tolerant to an attack A if $M \parallel A \sqsubseteq M$. We say that M is vulnerable to an attack A if there is a time interval $m..n$, with $m \in \mathbb{N}^+$ and $n \in \mathbb{N}^+ \cup \{\infty\}$, such that $M \parallel A \not\sqsubseteq_{m..n} M$.*

Thus, if a system M is vulnerable to an attack A of class $C \in [\mathcal{I} \rightarrow \mathcal{P}(m..n)]$, during the time interval $m'..n'$, then the attack operates during the interval $m..n$ but it influences the system under attack in the time interval $m'..n'$ (obviously, $m' \geq m$). If n' is finite, then we have a *temporary attack*, otherwise we have a *permanent attack*. Furthermore, if $m' - n$ is big enough and $n - m$ is small, then we have a quick nasty attack that affects the system late enough to allow *attack camouflages* [25]. On the other hand, if m' is significantly smaller than n , then the attack affects the observable behaviour of the system well before its termination and the CPS has good chances of undertaking countermeasures to stop the attack. Finally, if $M \parallel A \xrightarrow{t} \xrightarrow{\text{deadlock}}$, for some trace t , then we say that the attack A is *lethal*, as it is capable to halt (deadlock) the CPS M . This is obviously a permanent attack.

Note that, according to Definition 13, the tolerance (or vulnerability) of a CPS also depends on the capability of the *IDS* component to detect and signal undesired physical behaviours. In fact, the *IDS* component might be designed to detect abnormal physical behaviours going well further than deadlocks and violations of safety conditions.

According to the literature, we say that an attack is *stealthy* if it is able to drive the CPS under attack into an incorrect physical state (either deadlock or violation of the safety conditions) without being noticed by the *IDS* component.

4.1 Three different attacks on the physical devices of the CPS Sys

In this subsection, we present three different attacks to the CPS Sys described in Section 3. The formal proofs of the propositions stating the tolerance and/or the vulnerability of Sys with respect to these three attacks can be found in the associated technical report [39]. Here, we use Uppaal SMC to verify the models associated to the system under attack in order to detect deadlocks, violations of safety conditions, and IDS failures.

EXAMPLE 1. Consider the following DoS/Integrity attack on the the actuator cool, of class $C \in [\mathcal{I} \rightarrow \mathcal{P}(m..m)]$ with $C(\text{!cool?}) = C(\text{!cool!}) = \{m\}$ and $C(\iota) = \emptyset$, for $\iota \notin \{\text{!cool?}, \text{!cool!}\}$:

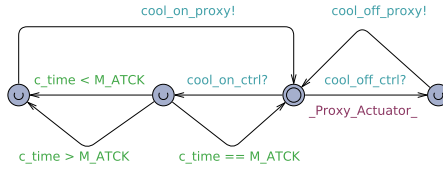
$$A_m = \text{tick}^{m-1} . \lfloor \text{drop cool}(x) . \text{if } (x=\text{off}) \{ \text{forge cool}(\text{off}) \} \text{ else } \{ \text{nil} \} \rfloor .$$

Here, the attack A_m operates exclusively in the m^{th} time slot, when it tries to drop an eventual cooling command (on or off) coming from the controller, and fabricates a fake command to turn off the cooling system. Thus, if the controller sends in the m^{th} time slot a command to turn off the coolant, then nothing bad happens as the attack will put the same message back. On the hand, if the controller sends a command to turn the cooling on, then the attack will drop the command. We recall that the controller will turn on the cooling only if the sensed temperature is greater than 10 (and hence $\text{temp} > 9.9$); this may happen only if $m > 8$. Since the command to turn the cooling on is never re-sent by Ctrl, the temperature will continue to rise, and after only 4 time units the system may violate the safety conditions emitting an action unsafe, while the IDS component will start sending alarms every 5 time units, until the whole system deadlocks because the temperature reaches the threshold of 50 degrees. Here, the IDS component of Sys is able to detect the attack with only one time unit delay.

PROPOSITION 3. Let Sys be our running example and A_m be the attack defined in Example 1. Then,

- $\text{Sys} \parallel A_m \sqsubseteq \text{Sys}$, for $1 \leq m \leq 8$,
- $\text{Sys} \parallel A_m \sqsubseteq_{m+4..∞} \text{Sys}$, for $m > 8$.

In order to support the statement of Proposition 3 we verify our Uppaal SMC model of Sys in which the communication network used by the controller to access the actuator is compromised. More precisely, we replace the *_Proxy_Actuator_* automaton of Figure 5 with a compromised one, provided in Figure 7, that implements the malicious activities of the MITM attacker A_m of Example 1.

Fig. 7. Uppaal SMC model for the attacker A_m of Example 1

We have done our analysis, with a 99% accuracy, for execution traces that are at most 1000 time units long and restricting the *attack time* m in the time interval 1..300. The results of our analysis are:

- when $m \in 1..8$, the attack is harmless as the system results to be safe, deadlock free and alarm free, with probability 0.99;
- when $m \in 9..300$, we have the following situation:
 - the probability that at the attack time m the controller sends a command to activate the cooling system (thus, triggering the attacker that will drop the command) can be obtained by verifying the property $\diamond_{[0,m]}(\text{Cooling_on} \wedge \text{global_clock} \geq m)$; as shown in Figure 8, when m grows in the time interval 1..300, the resulting probability stabilises around the value 0.096;
 - up to the $m+3^{\text{th}}$ time slot the system under attack remains safe, i.e., both properties $\square_{[1,m+3]}(\text{safe})$ and $\square_{[1,m+3]}(\neg \text{deadlock})$ hold with probability 0.99;
 - up to the $m+4^{\text{th}}$ time slot no alarms are fired, i.e., the property $\square_{[1,m+4]}(\neg \text{alarm})$ holds with probability 0.99 (no false positives);
 - in the $m+4^{\text{th}}$ time slot the system under attack might become unsafe as the probability, for $m \in 9..300$, that the property $\diamond_{[0,m+4]}(\neg \text{safe})$ is satisfied stabilises around the value 0.095;⁸
 - in the $m+5^{\text{th}}$ time slot the IDS may fire an alarm as the probability, for $m \in 9..300$, that the property $\diamond_{[0,m+5]}(\text{alarm})$ is satisfied stabilises around the value 0.094;⁹
 - the system under attack may deadlock as the property $\diamond_{[0,1000]}(\text{deadlocks})$ is satisfied with probability 0.096.¹⁰

EXAMPLE 2. Consider the following DoS/Integrity attack to the sensor s_t , of class $C \in [I \rightarrow \mathcal{P}(2..\infty)]$ such that $C(\zeta_{s_t}?) = \{2\}$, $C(\zeta_{s_t}!) = 2..\infty$ and $C(i) = \emptyset$, for $i \notin \{\zeta_{s_t}!, \zeta_{s_t}?\}$. The attack begins its activity in the time slot m , with $m > 8$, and then never stops:

$$\begin{aligned}
 A_m &= \text{tick}^{m-1}.A \\
 A &= \lfloor \text{sniff } s_t(x).\text{if } (x \leq 10) \{B(x)\} \text{ else } \{\text{tick}.A\} \rfloor \\
 B(y) &= \lfloor \text{forge } s_t\langle y \rangle.\text{tick}.B\langle y \rangle \rfloor B\langle y \rangle.
 \end{aligned}$$

Here, the attack A_m behaves as follows. It sleeps for $m - 1$ time slots and then, in the following time slot, it sniffs the current temperature at sensor s_t . If the sensed temperature v is greater than 10, then it moves to the next time slot and restarts sniffing; otherwise from that time on it will keep sending the same temperature v to the logical components (controller and IDS). Actually, once the forgery activity starts, the process Ctrl will always receive a temperature below 10 and will never activate the cooling

⁸Since this probability coincides with that of $\diamond_{[0,m]}(\text{Cooling_on} \wedge \text{global_clock} \geq m)$, it appears very likely that the activation of the cooling system in the m^{th} time slot triggers the attacker whose activity drags the system into an unsafe state with a delay of 4 time slots.

⁹As the two probabilities are pretty much the same, and $\square_{[1,m+3]}(\text{safe})$ and $\square_{[1,m+4]}(\neg \text{alarm})$ hold, the IDS seems to be quite effective in detecting the violations of the safety conditions in the $m+4^{\text{th}}$ time slot, with only one time slot delay.

¹⁰Since the probabilities are still the same, we argue that when the system reaches an unsafe state then it is not able to recover and it is doomed to deadlock.

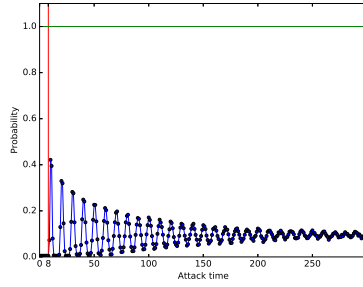


Fig. 8. Probability results of $\diamond_{[0,m]}(\text{Cooling_on} \wedge \text{global_clock} \geq m)$ by varying m in $1..300$

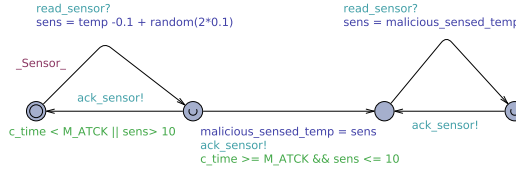


Fig. 9. Uppaal SMC model for the attacker A_m of Example 2

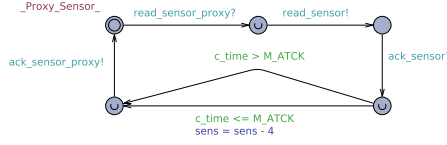
system (and consequently the IDS). As a consequence, the system under attack $\text{Sys} \parallel A$ will first move to an unsafe state until the invariant will be violated and the system will deadlock. Indeed, in the worst execution scenario, already in the $m+1^{\text{th}}$ time slot the temperature may exceed 10 degrees, and after 4 tick-actions, in the $m+5^{\text{th}}$ time slot, the system may violate the safety conditions emitting an unsafe action. Since the temperature will keep growing without any cooling activity, the deadlock of the CPS cannot be avoided. This is a lethal attack, as it causes a shut down of the system; it is also a stealthy attack as it remains undetected because the IDS never gets into action.

PROPOSITION 4. *Let Sys be our running example and A_m , for $m > 8$, be the attack defined in Example 2. Then $\text{Sys} \parallel A_m \sqsubseteq_{m+5..∞} \text{Sys}$.*

Here, we verify the Uppaal SMC model of Sys in which we assume that its sensor device is compromised (we recall that our MITM forgery attack on sensors or actuators can be assimilated to device compromise). In particular, we replace the $_Sensor_$ automaton of Figure 4 with a compromised one, provided in Figure 9, and implementing the malicious activities of the MITM attacker A_m of Example 2.

We have done our analysis, with a 99% accuracy, for execution traces that are at most 1000 time units long and restricting the *attack time* m in the integer interval $9..300$. The results of our analysis are:

- up to the $m+4^{\text{th}}$ time slot the system under attack remains safe, deadlock free, and alarm free, i.e., all three properties $\square_{[1,m+4]}(\text{safe})$, $\square_{[1,m+4]}(\neg\text{deadlock})$, and $\square_{[1,m+4]}(\neg\text{alarm})$ hold with probability 0.99;
- in the $m+5^{\text{th}}$ time slot the system under attack might become unsafe as the probability, for $m \in 9..300$, that the property $\diamond_{[0,m+5]}(\neg\text{safe})$ is satisfied stabilises around 0.104;
- the system under attack will eventually deadlock not later that 80 time slots after the attack time m , as the property $\square_{[m+80,1000]}(\text{deadlocks})$ is satisfied with probability 0.99;
- finally, the attack is stealthy as the property $\square_{[1,1000]}(\neg\text{alarm})$ holds with probability 0.99.

Fig. 10. Uppaal SMC model for the attacker A_n of Example 3

Now, let us examine a similar but less severe attack.

EXAMPLE 3. Consider the following DoS/Integrity attack to sensor s_t , of class $C \in [I \rightarrow \mathcal{P}(1..n)]$, with $C(\sharp s_t!) = C(\sharp s_t?) = 1..n$ and $C(\iota) = \emptyset$, for $\iota \notin \{\sharp s_t!, \sharp s_t?\}$:

$$\begin{aligned} A_n &= \lfloor \text{sniff } s_t(x). \lfloor \text{forge } s_t(x-4). \text{tick}. A_{n-1} \rfloor A_{n-1} \rfloor A_{n-1}, \text{ for } n > 0 \\ A_0 &= \text{nil}. \end{aligned}$$

In this attack, for n consecutive time slots, A_n sends to the logical components (controller and IDS) the current sensed temperature decreased by an offset 4. The effect of this attack on the system depends on the duration n of the attack itself: (i) for $n \leq 8$, the attack is harmless as the variable temp may not reach a (critical) temperature above 9.9; (ii) for $n = 9$, the variable temp might reach a temperature above 9.9 in the 9th time slot, and the attack would delay the activation of the cooling system of one time slot; as a consequence, the system might get into an unsafe state in the time interval 14..15, but no alarm will be fired; (iii) for $n \geq 10$, the system may get into an unsafe state in the time slot 14 and in the following $n + 11$ time slots; in this case, this would not be stealthy attack as the IDS will fire the alarm with a delay of at most two time slots later, rather this is a temporary attack that ends in the time slot $n + 11$.

PROPOSITION 5. Let Sys be our running example and A_n be the attack defined in Example 3. Then:

- $\text{Sys} \parallel A_n \sqsubseteq \text{Sys}$, for $n \leq 8$,
- $\text{Sys} \parallel A_n \sqsubseteq_{[14..15]} \text{Sys}$, for $n = 9$,
- $\text{Sys} \parallel A_n \sqsubseteq_{[14..n+11]} \text{Sys}$, for $n \geq 10$.

Here, we verify the Uppaal SMC model of Sys in which we replace the `_Proxy_Sensor_` automaton of Figure 5 with a compromised one, provided in Figure 10, and implementing the MITM activities of the attacker A_n of Example 3.

We have done our analysis, with a 99% accuracy, for execution traces that are at most 1000 time units long, and assuming that the *duration of the attack* n may vary in the integer interval 1..300. The results of our analysis are:

- when $n \in 1..8$, the system under attack remains safe, deadlock free, and alarm free, i.e., all three properties $\square_{[1,1000]}(\text{safe})$, $\square_{[1,1000]}(\neg \text{deadlock})$, and $\square_{[1,1000]}(\neg \text{alarm})$ hold with probability 0.99;
- when $n = 9$, we have the following situation:
 - the system under attack is deadlock free, i.e., the property $\square_{[1,1000]}(\neg \text{deadlock})$ holds with probability 0.99;
 - the system remains safe and alarm free, except for the time interval 14..15, i.e., all the following properties $\square_{[1,13]}(\text{safe})$, $\square_{[1,13]}(\neg \text{alarm})$, $\square_{[16,1000]}(\text{safe})$, and $\square_{[16,1000]}(\neg \text{alarm})$ hold with probability 0.99;
 - in the time interval 14..15, we may have violations of safety conditions, as the property $\diamond_{[0,14]}(\neg \text{safe} \wedge \text{global_clock} \geq 14)$ is satisfied with a probability 0.62, while the property $\diamond_{[0,15]}(\neg \text{safe} \wedge \text{global_clock} \geq 15)$ is satisfied with probability 0.21; both violations are *stealthy* as the property $\square_{[14,15]}(\neg \text{alarm})$ holds with probability 0.99;

- when $n \geq 10$, we have the following situation:
 - the system is deadlock free, i.e., the property $\Box_{[1,1000]}(\neg \text{deadlock})$ holds with probability 0.99;
 - the system remains safe except for the time interval $14..n+11$, i.e., the two properties $\Box_{[1,13]}(\text{safe})$ and $\Box_{[n+12,1000]}(\text{safe})$ hold with probability 0.99;
 - the system is alarm free except for the time interval $n+1..n+11$, i.e., the two properties $\Box_{[0,n]}(\neg \text{alarm})$ and $\Box_{[n+12,1000]}(\neg \text{alarm})$ hold with probability 0.99;
 - in the 14th time slot the system under attack may reach an unsafe state as the probability, for $n \in 10..300$, that the property $\Diamond_{[0,14]}(\neg \text{safe} \wedge \text{global_clock} \geq 14)$ is satisfied stabilises around 0.548;
 - once the attack has terminated, in the time interval $n+1..n+11$, the system under attack has good chances to reach an unsafe state as the probability, for $n \in 10..300$, that the property $\Diamond_{[0,n+11]}(\neg \text{safe} \wedge n+1 \leq \text{global_clock} \leq n+11)$ is satisfied stabilises around 0.672;
 - the violations of the safety conditions remain completely stealthy only up to the duration n of the attack (we recall that $\Box_{[0,n]}(\neg \text{alarm})$ is satisfied with probability 0.99); the probability, for $n \in 10..300$, that the property $\Diamond_{[0,n+11]}(\text{alarm})$ is satisfied stabilises around 0.13; thus, in the time interval $n+1..n+11$, only a small portion of violations of safety conditions are detected by the IDS while a great majority of them remains stealthy.

4.2 A technique for proving attack tolerance/vulnerability

In this subsection, we provide sufficient criteria to prove attack tolerance/vulnerability to attacks of an arbitrary class C . Actually, we do more than that: we provide sufficient criteria to prove attack tolerance/vulnerability to all attacks of any class C' that is somehow “weaker” than a given class C .

DEFINITION 14. Let $C_i \in [I \rightarrow \mathcal{P}(m_i..n_i)]$, for $i \in \{1, 2\}$, be two classes of attacks, with $m_1..n_1 \subseteq m_2..n_2$. We say that C_1 is weaker than C_2 , written $C_1 \leq C_2$, if $C_1(\iota) \subseteq C_2(\iota)$ for any $\iota \in I$.

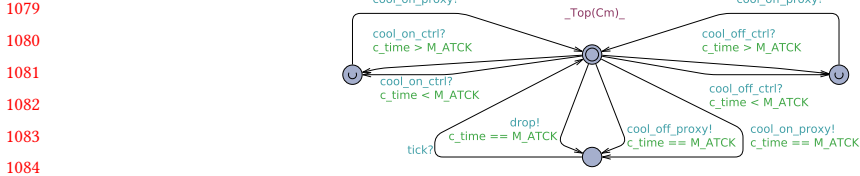
Intuitively, if $C_1 \leq C_2$ then: (i) the attacks of class C_1 might achieve fewer malicious activities than any attack of class C_2 (formally, there may be $\iota \in I$ such that $C_1(\iota) = \emptyset$ and $C_2(\iota) \neq \emptyset$); (ii) for those malicious activities $\iota \in I$ achieved by the attacks of both classes C_1 and C_2 (i.e., $C_1(\iota) \neq \emptyset$ and $C_2(\iota) \neq \emptyset$), if they may be perpetrated by the attacks of class C_1 at some time slot $k \in m_1..n_1$ (i.e., $k \in C_1(\iota)$) then all attacks of class C_2 may do the same activity ι at the same time k (i.e., $k \in C_2(\iota)$).

The next objective is to define a notion of *most powerful attack* (also called *top attacker*) of a given class C , such that, if a CPS M tolerates the most powerful attack of class C then it also tolerates *any* attack of class C' , with $C' \leq C$. We will provide a similar condition for attack vulnerability: let M be a CPS vulnerable to $\text{Top}(C)$ in the time interval $m_1..n_1$; then, for any attack A of class C' , with $C' \leq C$, if M is vulnerable to A then it is so for a smaller time interval $m_2..n_2 \subseteq m_1..n_1$.

Our notion of top attacker has two extra ingredients with respect to the physics-based attacks seen up to now: (i) *nondeterminism*, and (ii) time-unguarded recursive processes. This extra power of the top attacker is not a problem as we are looking for sufficient criteria.

For what concerns nondeterminism, we assume a generic procedure $\text{rnd}()$ that given an arbitrary set \mathcal{Z} returns an element of \mathcal{Z} chosen in a nondeterministic manner. This procedure allows us to express *nondeterministic choice*, $P \oplus Q$, as an abbreviation for the process $\text{if}(\text{rnd}(\{\text{true}, \text{false}\})) \{P\} \text{else} \{Q\}$. Thus, let $\iota \in \{\zeta p? : p \in \mathcal{S} \cup \mathcal{A}\} \cup \{\zeta p! : p \in \mathcal{S} \cup \mathcal{A}\}$, $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+ \cup \{\infty\}$, with $m \leq n$, and $\mathcal{T} \subseteq m..n$, we define the attack process $\text{Att}(\iota, k, \mathcal{T})$ ¹¹ as the attack which may achieve the malicious activity ι , at the time slot k , and which tries to do the same in all subsequent time slots of \mathcal{T} .

¹¹In case of sensor sniffing, we might avoid to add this specific attack process as our top attacker process can forge any possible value without need to read sensors.

Fig. 11. Uppaal SMC model for the top attacker $Top(C_m)$ of Example 4

Formally,

$$Att(\xi a?, k, \mathcal{T}) = \text{if } (k \in \mathcal{T}) \{ ([\text{drop } a(x)].Att(\xi a?, k, \mathcal{T})]Att(\xi a?, k+1, \mathcal{T}) \} \oplus \text{tick}.Att(\xi a?, k+1, \mathcal{T}) \\ \text{else } \{ \text{if } (k < \text{sup}(\mathcal{T})) \{ \text{tick}.Att(\xi a?, k+1, \mathcal{T}) \} \text{ else } \{ \text{nil} \} \}$$

$$Att(\xi s?, k, \mathcal{T}) = \text{if } (k \in \mathcal{T}) \{ ([\text{sniff } s(x)].Att(\xi s?, k, \mathcal{T})]Att(\xi s?, k+1, \mathcal{T}) \} \oplus \text{tick}.Att(\xi s?, k+1, \mathcal{T}) \\ \text{else } \{ \text{if } (k < \text{sup}(\mathcal{T})) \{ \text{tick}.Att(\xi s?, k+1, \mathcal{T}) \} \text{ else } \{ \text{nil} \} \}$$

$$Att(\xi p!, k, \mathcal{T}) = \text{if } (k \in \mathcal{T}) \{ ([\text{forge } p(\text{rnd}(\mathbb{R}))].Att(\xi p!, k, \mathcal{T})]Att(\xi p!, k+1, \mathcal{T}) \} \oplus \text{tick}.Att(\xi p!, k+1, \mathcal{T}) \\ \text{else } \{ \text{if } (k < \text{sup}(\mathcal{T})) \{ \text{tick}.Att(\xi p!, k+1, \mathcal{T}) \} \text{ else } \{ \text{nil} \} \}.$$

Note that, for $\mathcal{T} = \emptyset$, we assume $\text{sup}(\mathcal{T}) = -\infty$. We can now use the definition above to formalise the notion of most powerful attack of a given class C .

DEFINITION 15 (TOP ATTACKER). Let $C \in [I \rightarrow \mathcal{P}(m..n)]$ be a class of attacks. We define

$$Top(C) = \prod_{i \in I} Att(i, 1, C(i))$$

as the most powerful attack, or top attacker, of class C .

The following theorem provides soundness criteria for attack tolerance and attack vulnerability.

THEOREM 3 (SOUNDNESS CRITERIA). Let M be an honest and sound CPS, C an arbitrary class of attacks, and A an attack of a class C' , with $C' \leq C$.

- If $M \parallel Top(C) \sqsubseteq M$ then $M \parallel A \sqsubseteq M$.
- If $M \parallel Top(C) \sqsubseteq_{m_1..n_1} M$ then either $M \parallel A \sqsubseteq M$ or $M \parallel A \sqsubseteq_{m_2..n_2} M$, with $m_2..n_2 \subseteq m_1..n_1$.

COROLLARY 1. Let M be an honest and sound CPS, and C a class of attacks. If $Top(C)$ is not lethal for M then any attack A of class C' , with $C' \leq C$, is not lethal for M . If $Top(C)$ is not a permanent attack for M , then any attack A of class C' , with $C' \leq C$, is not a permanent attack for M .

The following example illustrates how Theorem 3 could be used to infer attack tolerance/vulnerability with respect to an entire class of attacks.

EXAMPLE 4. Consider our running example Sys and a class of attacks C_m , for $m \in \mathbb{N}$, such that $C_m(\xi cool?) = C_m(\xi cool!) = \{m\}$ and $C_m(i) = \emptyset$, for $i \notin \{\xi cool?, \xi cool!\}$. Attacks of class C_m may tamper with the actuator $cool$ only in the time slot m (i.e., in the time interval $m..m$). The attack A_m of Example 1 is of class C_m .

In the following analysis in Uppaal SMC of the top attacker $Top(C_m)$, we will show that both the vulnerability window and the probability of successfully attacking the system represent an upper bound for the attack A_m of Example 1 of class C_m . Technically, we verify the Uppaal SMC model of Sys in which we replace the $_Proxy_Actuator_$ automaton of Figure 5 with a compromised one, provided in Figure 11, and implementing the activities of the top attacker $Top(C_m)$. We carry out our analysis with a 99% accuracy, for execution traces that are at most 1000 time slots long, limiting the attack time m to the integer interval $1..300$.

1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176

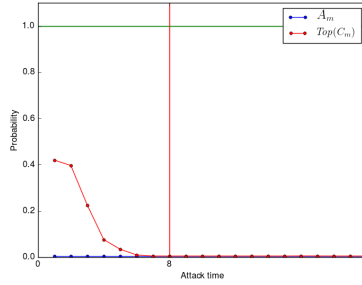


Fig. 12. Results of $\diamond_{[0, m+3]}(\text{deadlock} \wedge \text{global_clock} \geq m + 1)$ by varying the attack time m

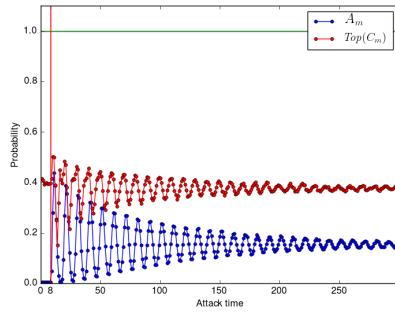


Fig. 13. Results of $\diamond_{[0, 1000]}(\text{deadlock} \wedge \text{global_clock} \geq m + 4)$ by varying the attack time m

To explain our analysis further, let us provide details on how $Top(C_m)$ affects Sys when compared to the attacker A_m of class C_m seen in the Example 1.

- In the time interval $1..m$, the attacked system remains safe, deadlock free, and alarm free. Formally, the three properties $\square_{[1, m]}(\text{safe})$, $\square_{[1, m]}(\neg \text{deadlock})$ and $\square_{[1, m]}(\neg \text{alarm})$ hold with probability 0.99. Thus, in this time interval, the top attacker is harmless, as well as A_m .
- In the time interval $m+1..m+3$, the system exposed to the top attacker may deadlock when $m \in 1..8$; for $m > 8$ the system under attack is deadlock free (see Figure 12). This is because the top attacker, unlike the attacker A_m , can forge in the first 8 time slots cool-on commands turning on the cooling and dropping the temperature below zero in the time interval $m+1..m+3$. Note that no alarms or unsafe behaviours occur in this case, as neither the safety process nor the IDS check whether the temperature drops below a certain threshold. Formally, the properties $\square_{[m+1, m+3]}(\text{safe})$ and $\square_{[m+1, m+3]}(\neg \text{alarm})$ hold with probability 0.99, as already seen for the attacker A_m .
- In the time interval $m+4..1000$, the top attacker has better chances to deadlock the system when compared with the attacker A_m (see Figure 13). For what concerns safety and alarms, the top attacker and the attacker A_m have the same probability of success (the properties $\square_{[m+4, 1000]}(\text{safe})$ and $\square_{[m+4, 1000]}(\neg \text{alarm})$ return the same probability results).

This example shows how the verification of a top attacker $Top(C)$ provides an upper bound of the effectiveness of the entire class of attacks C , in terms of both vulnerability window and

1177 probability of successfully attack the system. Of course, the accuracy of such approximation cannot
 1178 be estimated a priori.

1179

1180 5 IMPACT OF AN ATTACK

1181 In the previous section, we have grouped physics-based attacks by focussing on the physical
 1182 devices under attack and the timing aspects of the attack (Definition 12). Then, we have provided a
 1183 formalisation of when a CPS should be considered tolerant/vulnerable to an attack (Definition 13).
 1184 In this section, we show that these two formalisations are important not only to demonstrate
 1185 the tolerance (or vulnerability) of a CPS with respect to certain attacks, but also to evaluate the
 1186 disruptive impact of those attacks on the target CPS [22, 44].

1187 The goal of this section is to provide a *formal metric* to estimate the impact of a *successful attack*
 1188 on the *physical behaviour* of a CPS. In particular, we focus on the ability that an attack may have to
 1189 drag a CPS out of the correct behaviour modelled by its evolution map, with the given uncertainty.

1190 Recall that *evol* is *monotone* with respect to the uncertainty. Thus, as stated in Proposition 6, an
 1191 increase of the uncertainty may translate into a widening of the range of the possible behaviours
 1192 of the CPS. In the following, given the physical environment $E = \langle \text{evol}, \text{meas}, \text{inv}, \text{safe}, \xi_w, \xi_e \rangle$, we
 1193 write $E[\xi_w \leftarrow \xi'_w]$ as an abbreviation for $\langle \text{evol}, \text{meas}, \text{inv}, \text{safe}, \xi'_w, \xi_e \rangle$; similarly, for $M = E; S \bowtie P$
 1194 we write $M[\xi_w \leftarrow \xi'_w]$ for $E[\xi_w \leftarrow \xi'_w]; S \bowtie P$.

1195

1196 **PROPOSITION 6 (MONOTONICITY).** *Let M be an honest and sound CPS with uncertainty ξ_w . If*
 1197 $\xi_w \leq \xi'_w$ *and* $M \xrightarrow{t} M'$ *then* $M[\xi_w \leftarrow \xi'_w] \xrightarrow{t} M'[\xi_w \leftarrow \xi'_w]$.

1198

1199 However, a wider uncertainty in the model does not always correspond to a widening of the
 1200 possible behaviours of the CPS. In fact, this depends on the *intrinsic tolerance* of a CPS with respect
 1201 to changes in the uncertainty function. In the following, we will write $\xi_w + \xi'_w$ to denote the function
 1202 $\xi''_w \in \mathbb{R}^X$ such that $\xi''_w(x) = \xi_w(x) + \xi'_w(x)$, for any $x \in X$.

1203 **DEFINITION 16 (SYSTEM ξ -TOLERANCE).** *An honest and sound CPS M with uncertainty ξ_w is said*
 1204 ξ -*tolerant, for* $\xi \in \mathbb{R}^X$ *and* $\xi \geq 0$, *if*

1205

$$1206 \quad \xi = \sup \left\{ \xi' : M[\xi_w \leftarrow \xi_w + \eta] \sqsubseteq M, \text{ for any } 0 \leq \eta \leq \xi' \right\}.$$

1207

1208 Intuitively, if a CPS M has been designed with a given uncertainty ξ_w , but M is actually ξ -tolerant,
 1209 with $\xi > 0$, then the uncertainty ξ_w is somehow underestimated: the real uncertainty of M is given
 1210 by $\xi_w + \xi$. This information is quite important when trying to estimate the impact of an attack
 1211 on a CPS. In fact, if a system M has been designed with a given uncertainty ξ_w , but M is actually
 1212 ξ -tolerant, with $\xi > 0$, then an attack has (at least) a “room for manoeuvre” ξ to degrade the whole
 1213 CPS without being observed (and hence detected).

1214 Let Sys be our running example. In the rest of the section, with an abuse of notation, we will write
 1215 $\text{Sys}[\delta \leftarrow \gamma]$ to denote Sys where the uncertainty δ of the variable *temp* has been replaced with γ .

1216 **EXAMPLE 5.** *The CPS Sys is $\frac{1}{20}$ -tolerant as $\sup \left\{ \xi' : \text{Sys}[\delta \leftarrow \delta + \eta] \sqsubseteq \text{Sys}, \text{ for } 0 \leq \eta \leq \xi' \right\}$ is equal*
 1217 *to* $\frac{1}{20}$. *Since* $\delta + \xi = \frac{8}{20} + \frac{1}{20} = \frac{9}{20}$, *then this statement relies on the following proposition whose proof*
 1218 *can be found in the associated technical report [39].*

1219

1220 **PROPOSITION 7.** *We have*

1221

- 1222 • $\text{Sys}[\delta \leftarrow \gamma] \sqsubseteq \text{Sys}$, for $\gamma \in \left(\frac{8}{20}, \frac{9}{20} \right)$,
- 1223 • $\text{Sys}[\delta \leftarrow \gamma] \not\sqsubseteq \text{Sys}$, for $\gamma > \frac{9}{20}$.

1224 Now everything is in place to define our metric to estimate the impact of an attack.

1225

DEFINITION 17 (IMPACT). Let M be an honest and sound CPS with uncertainty ξ_w . We say that an attack A has definitive impact ξ on the system M if

$$\xi = \inf \left\{ \xi' : \xi' \in \mathbb{R}^X \wedge \xi' > 0 \wedge M \parallel A \sqsubseteq M[\xi_w \leftarrow \xi_w + \xi'] \right\}.$$

It has pointwise impact ξ on the system M at time m if

$$\xi = \inf \left\{ \xi' : \xi' \in \mathbb{R}^X \wedge \xi' > 0 \wedge M \parallel A \sqsubseteq_{m..n} M[\xi_w \leftarrow \xi_w + \xi'], n \in \mathbb{N}^+ \cup \{\infty\} \right\}.$$

Intuitively, the impact of an attacker A on a system M measures the perturbation introduced by the presence of the attacker in the compound system $M \parallel A$ with respect to the original system M . With this definition, we can establish either the definitive (and hence maximum) impact of the attack A on the system M , or the impact at a specific time m . In the latter case, by definition of $\sqsubseteq_{m..n}$, there are two possibilities: either the impact of the attack keeps growing after time m , or in the time interval $m+1$, the system under attack deadlocks.

The impact of $Top(C)$ provides an upper bound for the impact of all attacks of class C' , $C' \leq C$, as shown in the following theorem (proved in the appendix).

THEOREM 4 (TOP ATTACKER'S IMPACT). Let M be an honest and sound CPS, and C an arbitrary class of attacks. Let A be an arbitrary attack of class C' , with $C' \leq C$.

- The definitive impact of $Top(C)$ on M is greater than or equal to the definitive impact of A on M .
- If $Top(C)$ has pointwise impact ξ on M at time m , and A has pointwise impact ξ' on M at time m' , with $m' \leq m$, then $\xi' \leq \xi$.

In order to help the intuition on the impact metric defined in Definition 17, we give a couple of examples. Here, we focus on the role played by the size of the vulnerability window.

EXAMPLE 6. Let us consider the attack A_n of Example 3, for $n \in \{8, 9, 10\}$. Then,

- A_8 has definitive impact 0 on Sys,
- A_9 has definitive impact 0.23 on Sys,
- A_{10} has definitive impact 0.4 on Sys.

Formally, the impacts of these three attacks are obtained by calculating

$$\inf \{ \xi' : \xi' > 0 \wedge Sys \parallel A_n \sqsubseteq Sys[\delta \leftarrow \delta + \xi'] \},$$

for $n \in \{8, 9, 10\}$. Attack A_9 has a very low impact on Sys as it may drag the system into a temporary unsafe state in the time interval 14..15, whereas A_{10} has a slightly stronger impact as it may induce a temporary unsafe state during the larger time interval 14..21. Technically, since $\delta + \xi = 0.4 + 0.4 = 0.8$, the calculation of the impact of A_{10} relies on the following proposition whose proof can be found in the associated technical report [39].

PROPOSITION 8. Let A_{10} be the attack defined in Example 3. Then:

- $Sys \parallel A_{10} \not\sqsubseteq Sys[\delta \leftarrow \gamma]$, for $\gamma \in (0.4, 0.8)$,
- $Sys \parallel A_{10} \sqsubseteq Sys[\delta \leftarrow \gamma]$, for $\gamma > 0.8$.

On the other hand, the attack provided in Example 2, driving the system to a (permanent) deadlock state, has a much stronger impact on the CPS Sys than the attack of Example 3.

EXAMPLE 7. Let us consider the attack A_m of Example 2, for $m > 8$. As already discussed, this is a stealthy lethal attack that has a very severe and high impact. In fact, it has a definitive impact of 8.5 on the CPS Sys. Formally,

$$8.5 = \inf \left\{ \xi' : \xi' > 0 \wedge Sys \parallel A_m \sqsubseteq Sys[\delta \leftarrow \delta + \xi'] \right\}.$$

Technically, since $\delta + \xi = 0.4 + 8.5 = 8.9$, what stated in this example relies on the following proposition whose proof can be found in the associated technical report [39].

1275 PROPOSITION 9. Let A_m be the attack defined in Example 2. Then:

- 1276 • $\text{Sys} \parallel A_m \not\sqsubseteq \text{Sys}[\delta \leftarrow \gamma]$, for $\gamma \in (0.4, 8.9)$,
- 1277 • $\text{Sys} \parallel A_m \sqsubseteq \text{Sys}[\delta \leftarrow \gamma]$, for $\gamma > 8.9$.

1278 Thus, Definition 17 provides an instrument to estimate the impact of a *successful attack* on a CPS
 1279 in terms of the perturbation introduced both on its physical and on its logical processes. However,
 1280 there is at least another question that a CPS designer could ask: “Is there a way to estimate the
 1281 chances that an attack will be successful during the execution of my CPS?” To paraphrase in a
 1282 more operational manner: how many execution traces of my CPS are prone to be attacked by a
 1283 specific attack? As argued in the future work, we believe that *probabilistic metrics* might reveal to
 1284 be very useful in this respect [41].
 1285

1286 6 CONCLUSIONS, RELATED AND FUTURE WORK

1287 6.1 Summary

1288 We have provided *theoretical foundations* to reason about and *formally* detect attacks to physical
 1289 devices of CPSs. A straightforward utilisation of these methodologies is for *model-checking* or
 1290 *monitoring* in order to be able to formally analyse security properties of CPSs either before system
 1291 deployment or, when static analysis is not feasible, at runtime to promptly detect undesired
 1292 behaviours. To that end, we have proposed a hybrid process calculus, called CCPSA, as a formal
 1293 *specification language* to model physical and cyber components of CPSs as well as MITM physics-
 1294 based attacks. Note that our calculus is general enough to represent *Supervisory Control And Data*
 1295 *Acquisition (SCADA) systems as cyber components which can easily interact with controllers*
 1296 *and IDSs via channel communications. SCADA systems are the main technology used by system*
 1297 *engineers to supervise the activities of complex CPSs.*

1298 Based on CCPSA and its labelled transition semantics, we have formalised a threat model for CPSs
 1299 by grouping physics-based attacks in classes, according to the target physical devices and two timing
 1300 parameters: begin and duration of the attacks. Then, we developed two different *compositional*
 1301 trace semantics for CCPSA to assess *attack tolerance/vulnerability* with respect to a given attack.
 1302 Such a tolerance may hold *ad infinitum* or for a limited amount of time. In the latter case, the CPS
 1303 under attack is vulnerable and the attack affects the observable behaviour of the system only after
 1304 a certain point in time, when the attack itself may already be achieved or still working.

1305 Along the lines of GNDC [18], we have defined a notion of *top attacker*, $Top(C)$, of a given class of
 1306 attacks C , which has been used to provide sufficient criteria to prove attack tolerance/vulnerability
 1307 to all attacks of class C (and weaker ones).
 1308

1309 Then, we have provided a metric to estimate the *maximum impact* introduced in the system under
 1310 attack with respect to its genuine behaviour, according to its evolution law and the uncertainty
 1311 of the model. We have proved that the impact of the most powerful attack $Top(C)$ represents an
 1312 upper bound for the impact of any attack A of class C (and weaker ones).

1313 Finally, we have formalised a *running example* in Uppaal SMC [16], the statistical extension of
 1314 the Uppaal model checker [7]. Our goal was to test Uppaal SMC as an automatic tool for the *static*
 1315 *security analysis* of a simple but significant CPS exposed to a number of different physics-based
 1316 attacks with different impacts on the system under attack. Here, it is important to note that, although
 1317 we have verified most of the properties stated in the paper, we have not been able to capture time
 1318 properties on the responsiveness of the IDS to violations of the safety conditions. Examples of such
 1319 properties are: (i) there are time slots m and k such that the system may have an unsafe state at
 1320 some time $n > m$, and the IDS detects this violation with a delay of at least k time slots (k being a
 1321 lower bound of the reaction time of the IDS), or (ii) there is a time slot n in which the IDS fires an
 1322 alarm but neither an unsafe state nor a deadlock occurs in the time interval $n-k..n+k$: this would
 1323

1324 provide a tolerance of the occurrence of *false positive*. Furthermore, Uppaal SMC does not support
1325 the verification of nested formulae. Thus, although from a designer's point of view it would have
1326 been much more practical to verify a logic formula of the form $\exists\Diamond(\Box_{[t,t+5]}temp > 9.9)$ to check
1327 safety and invariant conditions, in Uppaal SMC we had to implement a `_Safety_` automaton that is
1328 not really part of our CPS (for more details see the discussion of related work).

1329

1330 6.2 Related work

1331 A number of approaches have been proposed for modelling CPSs using *hybrid process algebras* [8,
1332 15, 21, 52, 56]. Among these approaches, our calculus CCPSA shares some similarities with the
1333 ϕ -calculus [52]. However, unlike CCPSA, in the ϕ -calculus, given a hybrid system (E, P) , the process
1334 P can dynamically change the evolution law in E . Furthermore, the ϕ -calculus does not have a
1335 representation of physical devices and measurement law, which are instead crucial for us to model
1336 physics-based attacks that operate in a timely fashion on sensors and actuators. More recently,
1337 Galpin et al. [21] have proposed a process algebra in which the continuous part of the system
1338 is represented by appropriate variables whose changes are determined by active influences (i.e.,
1339 commands on actuators).

1340 Many good surveys on the security of cyber-physical systems have been published recently
1341 (see, e.g., [3, 24, 61, 62]), including a survey of surveys [23]. In particular, the surveys [61, 62]
1342 provide a systematic categorisation of 138 selected papers on CPS security. Among those 138
1343 papers, 65 adopt a discrete notion of time similar to ours, 26 a continuous one, 55 a quasi-static
1344 time model, and the rest use a hybrid time model. This study encouraged us in adopting a discrete
1345 time model for physical processes rather than a continuous one. *Still, one might wonder what is
1346 actually lost when one adopts a discrete rather than a continuous time model, in particular when
1347 the attacker has the possibility to move in a continuous time setting. A continuous time model
1348 is, of course, more expressive; for instance, Kanovich et al. [33] identified a novel vulnerability in
1349 the context of cryptographic protocols for CPSs in which the attacker works in a continuous-time
1350 setting to fool discrete-time verifiers. However, we believe that, for physics-based attacks, little is
1351 lost by adopting a discrete time model. In fact, sensor measurements and actuator commands are
1352 elaborated within controllers, which are digital devices with an intrinsic discrete notion of time. In
1353 particular, for what concerns dropping of actuator commands and forging of sensor measurements,
1354 there are no differences between discrete-time and continuous-time attackers given that to achieve
1355 those malicious activities the attacker has to synchronise with the controller. Thus, there remain
1356 only two potential malicious activities: sensor sniffing and forging of actuator commands. Can a
1357 continuous-time attacker, able to carry out these two malicious activities, be more disruptive than
1358 a similar attacker adopting a discrete-time model? This would only be the case when dealing with
1359 very rare physical processes changing their physical state in an extremely fast way, faster than
1360 the controller which is the one dictating the discrete time of the CPS. However, we believe that
1361 CPSs of this kind would hardly be controllable as they would pose serious safety issues even in the
1362 absence of any attacker.*

1363 The survey [24] provides an exhaustive review of papers on physics-based anomaly detection
1364 proposing a unified taxonomy, whereas the survey [3] presents the main solutions in the estimation
1365 of the consequences of cyber-attacks, attacks modelling and detection, and the development of
1366 security architecture (the main types of attacks and threats against CPSs are analysed and grouped
1367 in a tree structure).

1368 Huang et al. [31] were among the first to propose *threat models* for CPSs. Along with [34, 35],
1369 they stressed the role played by timing parameters on integrity and DoS attacks.

1370 Gollmann et al. [25] discussed possible goals (*equipment damage, production damage, compliance
1371 violation*) and stages (*access, discovery, control, damage, cleanup*) of physics-based attacks. In this
1372

1372

1373 article, we focused on the “damage” stage, where the attacker already has a rough idea of the
1374 plant and the control architecture of the target CPS. As we remarked in Section 1, we here focus
1375 on an attacker who has already entered the CPS, without considering how the attacker gained
1376 access to the system, which could have happened in several ways, for instance by attacking an
1377 Internet-accessible controller or one of the communication protocols.

1378 Almost all papers discussed in the surveys mentioned above [3, 24, 62] investigate attacks on
1379 CPSs and their protection by relying on *simulation test systems* to validate the results, rather than
1380 *formal methodologies*. We are aware of a number of works applying *formal methods* to CPS security,
1381 although they apply methods, and most of the time have goals, that are quite different from ours.
1382 We discuss the most significant ones on the following.

1383 Burmester et al. [12] employed *hybrid timed automata* to give a threat framework based on the
1384 traditional Byzantine faults model for crypto-security. However, as remarked in [54], physics-based
1385 attacks and faults have inherently distinct characteristics. Faults are considered as physical events
1386 that affect the system behaviour where simultaneous events don’t act in a coordinated way, whereas
1387 cyber attacks may be performed over a significant number of attack points and in a coordinated way.

1388 In [58], Vigo presented an attack scenario that addresses some of the peculiarities of a cyber-
1389 physical adversary, and discussed how this scenario relates to other attack models popular in the
1390 security protocol literature. Then, in [59] Vigo et al. proposed an untimed calculus of broadcasting
1391 processes equipped with notions of failed and unwanted communication. These works differ quite
1392 considerably from ours, e.g., they focus on *DoS attacks* without taking into consideration timing
1393 aspects or impact of the attack.

1394 Cómbita et al. [14] and Zhu and Basar [63] applied *game theory* to capture the conflict of goals
1395 between an attacker who seeks to maximise the damage inflicted to a CPS’s security and a defender
1396 who aims to minimise it [43].

1397 Rocchetto and Tippenhaur [51] introduced a taxonomy of the diverse attacker models proposed
1398 for CPS security and outline requirements for generalised attacker models; in [50], they then
1399 proposed an extended *Dolev-Yao attacker model* suitable for CPSs. In their approach, physical layer
1400 interactions are modelled as abstract interactions between logical components to support reasoning
1401 on the physical-layer security of CPSs. This is done by introducing additional orthogonal channels.
1402 Time is not represented.

1403 Nigam et al. [46] worked around the notion of *Timed Dolev-Yao Intruder Models for Cyber-Physical*
1404 *Security Protocols* by bounding the number of intruders required for the automated verification of
1405 such protocols. Following a tradition in security protocol analysis, they provide an answer to the
1406 question: How many intruders are enough for verification and where should they be placed? They
1407 also extend the strand space model to CPS protocols by allowing for the symbolic representation of
1408 time, so that they can use the tool Maude [47] along with SMT support. Their notion of time is
1409 however different from ours, as they focus on the time a message needs to travel from an agent to
1410 another. The paper does not mention physical devices, such as sensors and/or actuators.

1411 There are a few approaches that carry out *information flow security analysis* on discrete/continuous
1412 models for CPSs. Akella et al. [2] proposed an approach to perform information flow analysis,
1413 including both trace-based analysis and automated analysis through process algebra specification.
1414 This approach has been used to verify process algebra models of a gas pipeline system and a smart
1415 electric power grid system. Bodei et al. [10] proposed a process calculus supporting a control flow
1416 analysis that safely approximates the abstract behaviour of IoT systems. Essentially, they track
1417 how data spread from sensors to the logics of the network, and how physical data are manipulated.
1418 In [9], the same authors extend their work to infer *quantitative measures* to establish the cost of
1419 possibly security countermeasures, in terms of time and energy. Another discrete model has been
1420 proposed by Wang [60], where Petri-net models have been used to verify *non-deducibility security*
1421

1422 *properties* of a natural gas pipeline system. More recently, Bohrer and Platzer [11] introduced dHL,
1423 a hybrid logic for verifying cyber-physical hybrid-dynamic information flows, communicating
1424 information through both discrete computation and physical dynamics, so security is ensured even
1425 when attackers observe *continuously-changing values* in continuous time.

1426 Huang et al. [30] proposed a *risk assessment method* that uses a Bayesian network to model the
1427 attack propagation process and infers the probabilities of sensors and actuators to be compromised.
1428 These probabilities are fed into a stochastic hybrid system (SHS) model to predict the evolution of
1429 the physical process being controlled. Then, the security risk is quantified by evaluating the system
1430 availability with the SHS model.

1431 As regards tools for the formal verification of CPSs, we remark that we tried to verify our case
1432 study using *model-checking tools* for distributed systems such as PRISM [37], Uppaal [6], Real-Time
1433 Maude [47], and *prohver* within the MODEST TOOLSET [27]. In particular, as our example adopts a
1434 discrete notion of time, we started looking at tools supporting discrete time. PRISM, for instance,
1435 relies on Markov decision processes or discrete-time Markov chains, depending on whether one is
1436 interested in modelling nondeterminism or not. It supports the verification of both CTL and LTL
1437 properties (when dealing with nonprobabilistic systems). This allowed us to express the formula
1438 $\exists\Diamond(\Box_{[t,t+5]}temp > 9.9)$ to verify violations of the safety conditions, avoiding the implementation
1439 of the *_Safety_* automaton. However, using integer variables to represent state variables with a
1440 fixed precision requires the introduction of extra transitions (to deal with nondeterministic errors),
1441 which significantly complicates the PRISM model. In this respect, Uppaal appears to be more
1442 efficient than PRISM, as we have been able to concisely express the error occurring in integer
1443 state variables thanks to the *select()* construct, in which the user can fix the granularity adopted to
1444 approximate a dense interval. This discrete representation provides an *under-approximation* of the
1445 system behaviour; thus, a finer granularity translates into an exponential increase of the complexity
1446 of the system, with obvious consequences on the verification performance. Then, we tried to model
1447 our case study in Real-Time Maude, a completely different framework for real-time systems, based
1448 on *rewriting logic*. The language supports object-like inheritance features that are quite helpful to
1449 represent complex systems in a modular manner. We use communication channels to implement our
1450 attacks on the physical devices. Furthermore, we used rational variables for a more concise discrete
1451 representation of state variables. We have been able to verify LTL and T-CTL properties, although
1452 the verification process resulted to be quite slow due to a proliferation of rewriting rules when fixing
1453 a reasonable granularity to approximate dense intervals. As the verification logic is quite powerful,
1454 there is no need to implement an ad-hoc process to check for safety. Finally, we also tried to model
1455 our case study in the *safety model checker* *prohver* within the MODEST TOOLSET. We specified our case
1456 study in the high-level language HMODEST, supporting: (i) differential inclusion to model linear CPSs
1457 with constant bounded derivatives; (ii) linear formulae to express nondeterministic assignments
1458 within a dense interval; (iii) a compositional programming style inherited from process algebra;
1459 (iv) shared actions to synchronise parallel components. However, we faced the same performance
1460 limitations encountered in Uppaal. Thus, we decided to move to statistical model checking.

1461 Finally, this article extends the preliminary conference version [40] in the following aspects: (i) the
1462 calculus has been slightly redesigned by distinguishing physical state and physical environment,
1463 adding specifying constructs to sniff, drop and forge packets, and removing, for simplicity, protected
1464 physical devices; (ii) the two trace semantics have been proven to be compositional, i.e., preserved
1465 by properly defined contexts; (iii) both our running example *Sys* and the attacks proposed in
1466 Examples 1, 2, 3 and 4 have been implemented and verified in Uppaal SMC.

1467
1468
1469
1470

6.3 Future work

While much is still to be done, we believe that our paper provides a stepping stone for the development of formal and automated tools to analyse the security of CPSs. We will consider applying, possibly after proper enhancements, existing tools and frameworks for automated security protocol analysis, resorting to the development of a dedicated tool if existing ones prove not up to the task. We will also consider further security properties and concrete examples of CPSs, as well as other kinds of physics-based attacks, *such as delays in the communication of measurements and/or commands, and periodic attacks, i.e., attacks that operate in a periodic fashion inducing periodic physical effects on the targeted system that may be easily confused by engineers with system malfunctions*. This will allow us to refine the classes of attacks we have given here (e.g., by formalising a type system amenable to static analysis), and provide a formal definition of when a CPS is more secure than another so as to be able to design, by progressive refinement, secure variants of a vulnerable CPSs.

We also aim to extend the behavioural theory of CCPSA by developing suitable *probabilistic metrics* to take into consideration the probability of a specific trace to actually occur. We have already done some progress in this direction for a variant of CCPSA with no security features in it, by defining ad hoc compositional *bisimulation metrics* [42]. In this manner, we believe that our notion of impact might be refined by taking into account quantitative aspects of an attack such as the probability of being successful when targeting a specific CPS. A first attempt on a (much) simpler IoT setting can be found in [41].

Finally, for what concerns automatic approximations of the impact, while we have not yet fully investigated the problem, we believe that we can transform it into a “minimum problem”. For instance, if the environment uses linear functions, then, by adapting techniques developed for linear hybrid automata (see, e.g., [4]), the set of all traces with length at most n (for a fixed n) can be characterised by a system of first degree inequalities, so the measure of the impact could be translated into a linear programming problem.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful and careful reviews.

REFERENCES

- [1] Guide to increased security in industrial information and control systems. Swedish Civil Contingencies Agency, 2014.
- [2] R. Akella, H. Tang, and B. M. McMillin. Analysis of information flow security in cyber-physical systems. *IJCIP*, 3(3-4):157–173, 2010.
- [3] R. M. Alguliyev, Y. Imamverdiyev, and L. Sukhostat. Cyber-physical systems and their security issues. *Computers in Industry*, 100:212–223, 2018.
- [4] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *TCS*, 138(1):3–34, 1995.
- [5] E. Bartocci, J. V. Deshmukh, A. Donzè, G. E. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS*, pages 135–175. Springer, 2018.
- [6] G. Behrmann, A. David, L. K. G., J. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. UPPAAL 4.0. In *QEST 2006*, pages 125–126. IEEE Computer Society, 2006.
- [7] G. Behrmann, A. David, and K. Larsen. A tutorial on Uppaal. In *International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Revised Lectures*, pages 200–236, 2004.
- [8] J. A. Bergstra and C. A. Middleburg. Process algebra for hybrid systems. *TCS*, 335(2-3):215–280, 2005.
- [9] C. Bodei, S. Chessa, and L. Galletta. Measuring security in IoT communications. *TCS*, to appear.
- [10] C. Bodei, P. Degano, G. Ferrari, and L. Galletta. Tracing where IoT data are collected and aggregated. *Logical Methods in Computer Science*, 13(3):1–38, 2017.
- [11] B. Bohrer and A. Platzer. A Hybrid, Dynamic Logic for Hybrid-Dynamic Information Flow. In *LICS*, pages 115–124. ACM, 2018.

- 1520 [12] M. Burmester and E. Magkos. Modeling security in cyber-physical systems. *IJCIP*, 5(3-4):118–126, 2012.
- 1521 [13] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 12 1952.
- 1522 [14] L. F. Cómbita, J. Giraldo, A. A. Cárdenas, and N. Quijano. Response and reconfiguration of cyber-physical control systems: A survey. In *CCAC*, pages 1–6. IEEE, 2015.
- 1523 [15] P. Cuijpers and M. Reniers. Hybrid process algebra. *JLAP*, 62(2):191–245, 2005.
- 1524 [16] A. David, K. G. Larsen, A. Legay, M. Mikušionis, and D. B. Poulsen. Uppaal SMC tutorial. *STTT*, 17(4):397–415, 2015.
- 1525 [17] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, (2):198–208, 1983.
- 1526 [18] R. Focardi and F. Martinelli. A Uniform Approach for the Definition of Security Properties. In *FM*, volume 1708 of *LNCS*, pages 794–813. Springer, 1999.
- 1527 [19] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. *STTT*, 10(3):263–279, 2008.
- 1528 [20] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*, volume 6806 of *LNCS*, pages 379–395. Springer, 2011.
- 1529 [21] V. Galpin, L. Bortolussi, and J. Hillston. HYPE: Hybrid modelling by composition of flows. *FOAC*, 25(4):503–541, 2013.
- 1530 [22] B. Genge, I. Kiss, and P. Haller. A system dynamics approach for assessing the impact of cyber attacks on critical infrastructures. *Int. J. Critical Infrastructure Protection*, 10:3–17, 2015.
- 1531 [23] J. Giraldo, E. Sarkar, A. A. Cárdenas, M. Maniatakos, and M. Kantarcioglu. Security and Privacy in Cyber-Physical Systems: A Survey of Surveys. *IEEE Design & Test*, 34(4):7–17, 2017.
- 1532 [24] J. Giraldo, D. I. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell. A Survey of Physics-Based Attack Detection in Cyber-Physical Systems. *ACM Comput. Surv.*, 51(4):76:1–76:36, 2018.
- 1533 [25] 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 *ACM CCPS*, pages 1–12, 2015.
- 1534 [26] D. Gollmann and M. Krotofil. Cyber-Physical Systems Security. In *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, volume 9100 of *LNCS*, pages 195–204. Springer, 2016.
- 1535 [27] A. Hartmanns and H. Hermanns. The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification. In *TACAS 2014*, volume 8413 of *LNCS*, pages 593–598. Springer, 2014.
- 1536 [28] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, 1995.
- 1537 [29] T. A. Henzinger, P. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *STTT*, 1(1-2):110–122, 1997.
- 1538 [30] K. Huang, C. Zhou, Y. Tian, S. Yang, and Y. Qin. Assessing the Physical Impact of Cyberattacks on Industrial Cyber-Physical Systems. *IEEE Trans. Industrial Electronics*, 65(10):8153–8162, 2018.
- 1539 [31] Y. Huang, A. A. Cárdenas, S. Amin, Z. Lin, H. Tsai, and S. Sastry. Understanding the physical and economic consequences of attacks on control systems. *IJCIP*, 2(3):73–83, 2009.
- 1540 [32] ICS-CERT. Cyber-Attack Against Ukrainian Critical Infrastructure. <https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01>.
- 1541 [33] M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. L. Talcott. Discrete vs. Dense Times in the Analysis of Cyber-Physical Security Protocols. In *POST*, volume 9036 of *LNCS*, pages 259–279. Springer, 2015.
- 1542 [34] M. Krotofil and A. A. Cárdenas. Resilience of Process Control Systems to Cyber-Physical Attacks. In *NordSec*, volume 8208 of *LNCS*, pages 166–182. Springer, 2013.
- 1543 [35] M. Krotofil, A. A. Cárdenas, J. Larsen, and D. Gollmann. Vulnerabilities of cyber-physical systems to stale data - Determining the optimal time to launch attacks. *Int. J. Critical Infrastructure Protection*, 7(4):213–232, 2014.
- 1544 [36] D. Kushner. The real story of STUXnet. *IEEE Spectrum*, 50(3):48 – 53, 2013.
- 1545 [37] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.
- 1546 [38] R. Lanotte and M. Merro. A Calculus of Cyber-Physical Systems. In *LATA*, volume 10168 of *LNCS*, pages 115–127. Springer, 2017.
- 1547 [39] R. Lanotte, M. Merro, A. Munteanu, and L. Viganò. A Formal Approach to Physics-Based Attacks in Cyber-Physical Systems (Extended Version). *CoRR*, abs/1902.04572, 2019.
- 1548 [40] R. Lanotte, M. Merro, R. Muradore, and L. Viganò. A Formal Approach to Cyber-Physical Attacks. In *CSF*, pages 436–450. IEEE Computer Society, 2017.
- 1549 [41] R. Lanotte, M. Merro, and S. Tini. Towards a Formal Notion of Impact Metric for Cyber-Physical Attacks. In *IFM*, volume 11023 of *LNCS*, pages 296–315. Springer, 2018.
- 1550 [42] R. Lanotte, M. Merro, and S. Tini. A Probabilistic Calculus of Cyber-Physical Systems. *I&C*, 2019.
- 1551 [43] M. Manshaei, Q. Zhu, T. Alpcan, T. Basar, and J.-P. Hubaux. Game theory meets network security and privacy. *ACM Computer Surveys*, 45(3):25, 2013.
- 1552 [44] J. Milosevic, D. Umsonst, H. Sandberg, and K. H. Johansson. Quantifying the Impact of Cyber-Attack Strategies for Control Systems Equipped With an Anomaly Detector. In *ECC*, pages 331–337. IEEE, 2018.

- 1569 [45] A. F. Murillo-Piedrahita, V. Gaur, J. Giraldo, A. A. Cárdenas, and S. J. Rueda. Virtual incident response functions in
 1570 control systems. *Computer Networks*, 135:147–159, 2018.
- 1571 [46] V. Nigam, C. Talcott, and A. A. Urquiza. Towards the Automated Verification of Cyber-Physical Security Protocols:
 1572 Bounding the Number of Timed Intruders. In *ESORICS*, volume 9879 of *LNCS*, pages 450–470. Springer, 2016.
- 1573 [47] P. C. Ölveczky and J. Meseguer. Semantics and pragmatics of Real-Time Maude. *Higher-Order and Symbolic Computation*,
 1574 20(1-2):161–196, 2007.
- 1575 [48] A. Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, 2018.
- 1576 [49] J. Quesel, S. Mitsch, S. M. Loos, N. Arechiga, and A. Platzer. How to model and prove hybrid systems with KeYmaera:
 1577 a tutorial on safety. *STTT*, 18(1):67–91, 2016.
- 1578 [50] M. Rocchetto and N. O. Tippenhauer. CPDY: Extending the Dolev-Yao Attacker with Physical-Layer Interactions. In
 1579 *ICFEM*, volume 10009 of *LNCS*, pages 175–192, 2016.
- 1580 [51] M. Rocchetto and N. O. Tippenhauer. On Attacker Models and Profiles for Cyber-Physical Systems. In *ESORICS*,
 1581 volume 9879 of *LNCS*, pages 427–449. Springer, 2016.
- 1582 [52] W. C. Rounds and H. Song. The ϕ -calculus: A language for distributed control of reconfigurable embedded systems. In
 1583 *HSCC*, volume 2623 of *LNCS*, pages 435–449. Springer, 2003.
- 1584 [53] J. Slay and M. Miller. Lessons Learned from the Maroochy Water Breach. In *Critical Infrastructure Protection*, volume
 1585 253 of *IFIP*, pages 73–82. Springer, 2007.
- 1586 [54] A. Teixeira, I. Shames, J. Sandberg, and K. H. Johansson. A secure control framework for resource-limited adversaries.
 1587 *Automatica*, 51:135–148, 2015.
- 1588 [55] U.S. Chemical Safety and Hazard Investigation Board, T2 Laboratories Inc. Reactive Chemical Explosion: Final
 1589 Investigation Report. Report No. 2008-3-I-FL, 2009.
- 1590 [56] D. van Beek, K. Man, M. Reniers, J. Rooda, and R. Schiffelers. Syntax and consistent equation semantics of hybrid chi.
 1591 *The Journal of Logic and Algebraic Programming*, 68(1&2):129–210, 2006.
- 1592 [57] P. Vasilikos, F. Nielson, and H. R. Nielson. Secure Information Release in Timed Automata. In *POST*, volume 10804 of
 1593 *Lecture Notes in Computer Science*, pages 28–52. Springer, 2018.
- 1594 [58] R. Vigo. The Cyber-Physical Attacker. In *SAFECOMP*, volume 7613 of *LNCS*, pages 347–356. Springer, 2012.
- 1595 [59] R. Vigo, F. Nielson, and H. Riis Nielson. Broadcast, denial-of-service, and secure communication. In *IFM*, volume 7940
 1596 of *LNCS*, pages 412–427. Springer, 2013.
- 1597 [60] J. Wang and H. Yu. Analysis of the Composition of Non-Deducibility in Cyber-Physical Systems. *Applied Mathematics
 1598 & Information*, 8:3137–3143, 2014.
- 1599 [61] Y. Zaccchia Lun, A. D’Innocenzo, I. Malavolta, and M. D. Di Benedetto. Cyber-Physical Systems Security: a Systematic
 1600 Mapping Study. *CoRR*, abs/1605.09641, 2016.
- 1601 [62] Y. Zaccchia Lun, A. D’Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto. State of the art of cyber-physical
 1602 systems security: An automatic control perspective. *Journal of Systems and Software*, 149:174–216, 2019.
- 1603 [63] Q. Zhu and T. Basar. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems:
 1604 games-in-games principle for optimal cross-layer resilient control systems. *IEEE Control Systems*, 35(1):46–65, 2015.

1605 A PROOFS

1606 As already stated in Remark 2, our trace preorder \sqsubseteq is deadlock-sensitive. Formally,

1607 LEMMA 1. *Let M and N be two CPSs in CCPSA such that $M \sqsubseteq N$. Then, M satisfies its system
 1608 invariant if and only if N satisfies its system invariant.*

1609 PROOF. This is because CPSs that don’t satisfy their invariant can only fire deadlock actions. \square

1610 PROOF OF THEOREM 1. We prove the three statements separately.

1611 (1) Let us prove that $M \uplus O \xrightarrow{t} M' \uplus O'$ entails $N \uplus O \xrightarrow{\hat{t}} N' \uplus O'$. The proof is by induction
 1612 on the length of the trace $M \uplus O \xrightarrow{t} M' \uplus O'$.

1613 As $M \sqsubseteq N$, by an application of Lemma 1 it follows that either both M and N satisfy their
 1614 respective invariants or they both don’t. In the latter case, the result would be easy to prove
 1615 as the systems can only fire deadlock actions. Similarly, if the system invariant of O is not
 1616 satisfied, then $M \uplus O$ and $N \uplus O$ can perform only deadlock actions and again the result would
 1617 follow easily. Thus, let us suppose that the system invariants of M , N and O are satisfied.

Base case. We suppose $M = E_1; S_1 \bowtie P_1$, $N = E_2; S_2 \bowtie P_2$, and $O = E_3; S_3 \bowtie P_3$. We proceed by case analysis on why $M \uplus O \xrightarrow{\alpha} M' \uplus O'$, for some action α .

• $\alpha = \bar{c}v$. Suppose $M \uplus O \xrightarrow{\bar{c}v} M' \uplus O'$ is derived by an application of rule (Out). We have two possible cases:

- either $P_1 \parallel P_3 \xrightarrow{\bar{c}v} P_1 \parallel P'_3$, because $P_3 \xrightarrow{\bar{c}v} P'_3$, for some P'_3 , $O' = S_3 \bowtie P'_3$, and $M' = M$,
- or $P_1 \parallel P_3 \xrightarrow{\bar{c}v} P'_1 \parallel P_3$, because $P_1 \xrightarrow{\bar{c}v} P'_1$, for some P'_1 , and $M = S_1 \bowtie P'_1$ and $O' = O$.

In the first case, by an application of rule (Par) we derive $P_2 \parallel P_3 \xrightarrow{\bar{c}v} P_2 \parallel P'_3$. Since both system invariants of N and O are satisfied, we can derive the required trace $N \uplus O \xrightarrow{\bar{c}v} N \uplus O'$ by an application of rule (Out). In the second case, since $P_1 \xrightarrow{\bar{c}v} P'_1$ and the invariant of M is satisfied, by an application of rule (Out) we can derive $M \xrightarrow{\bar{c}v} M'$. As $M \sqsubseteq N$, there exists a trace $N \xrightarrow{\bar{c}v} N'$, for some system N' . Thus, by several applications of rule (Par) we can easily derive $N \uplus O \xrightarrow{\bar{c}v} N' \uplus O = N' \uplus O'$, as required.

• $\alpha = cv$. Suppose $M \uplus O \xrightarrow{cv} M' \uplus O'$ is derived by an application of rule (Inp). This case is similar to the previous one.

• $\alpha = \tau$. Suppose $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (SensRead). We have two possible cases:

- either $P_1 \parallel P_3 \xrightarrow{s?v} P_1 \parallel P'_3$ because $P_3 \xrightarrow{s?v} P'_3$, for some P'_3 , $P_1 \parallel P_3 \xrightarrow{\not{s!v}}$ (and hence $P_3 \xrightarrow{\not{s!v}}$), $M' = M$ and $O' = S_3 \bowtie P'_3$,
- or $P_1 \parallel P_3 \xrightarrow{s?v} P'_1 \parallel P_3$ because $P_1 \xrightarrow{s?v} P'_1$, for some P'_1 , $P_1 \parallel P_3 \xrightarrow{\not{s!v}}$ (and hence $P_1 \xrightarrow{\not{s!v}}$) and $M' = S_1 \bowtie P'_1$ and $O' = O$.

In the first case, by an application of rule (Par) we derive $P_2 \parallel P_3 \xrightarrow{s?v} P_2 \parallel P'_3$. Moreover from $P_3 \xrightarrow{\not{s!v}}$ and since the sets of sensors are always disjoint, we can derive $P_2 \parallel P_3 \xrightarrow{\not{s!v}}$. Since both invariants of N and O are satisfied, we can derive $N \uplus O \xrightarrow{\tau} N \uplus O'$ by an application of rule (SensRead), as required. In the second case, since $P_1 \xrightarrow{s?v} P'_1$ and the invariant of M is satisfied, by an application of rule (SensRead) we can derive $M \xrightarrow{\tau} M'$ with $M' = S_1 \bowtie P'_1$. As $M \sqsubseteq N$, there exists a derivation $N \xrightarrow{\hat{\tau}} N'$, for some N' . Thus, we can derive the required trace $N \uplus O \xrightarrow{\hat{\tau}} N' \uplus O$ by an application of rule (Par).

• $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule ($\not{s}SensSniff\hat{s}$). This case is similar to the previous one.

• $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (ActWrite). This case is similar to the case (SensRead).

• $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule ($\not{s}AIntegr\hat{s}$). This case is similar to the case ($\not{s}SensSniff\hat{s}$).

• $\alpha = \tau$. Suppose that $M \uplus O \xrightarrow{\tau} M' \uplus O'$ is derived by an application of rule (Tau). We have four possible cases:

- $P_1 \parallel P_3 \xrightarrow{\tau} P'_1 \parallel P'_3$ by an application of rule (Com). We have two sub-cases: either $P_1 \xrightarrow{\bar{c}v} P'_1$ and $P_3 \xrightarrow{cv} P'_3$, or $P_1 \xrightarrow{cv} P'_1$ and $P_3 \xrightarrow{\bar{c}v} P'_3$, for some P'_1 and P'_3 . We prove the first case, the second one is similar. As the invariant of M is satisfied, by an

application of rule (Out) we can derive $M \xrightarrow{\bar{c}v} M'$. As $M \sqsubseteq N$, there exists a trace $N \xRightarrow{\hat{t}} \xrightarrow{\bar{c}v} \xRightarrow{\hat{t}} N'$, for some $N' = E_2; S'_2 \bowtie P'_2$. As $P_3 \xrightarrow{cv} P'_3$, by several applications

of rule (Par) and one of rule (Com) we derive $N \uplus O \xRightarrow{\bar{c}v} N' \uplus O'$, as required.

- $P_1 \parallel P_3 \xrightarrow{\tau} P_1 \parallel P'_3$ or $P_1 \parallel P_3 \xrightarrow{\tau} P'_1 \parallel P_3$ by an application of (Par). This case is easy.
- $P_1 \parallel P_3 \xrightarrow{\tau} P'_1 \parallel P'_3$ by an application of either rule (ℓ ActDrop ℓ) or rule (ℓ SensIntegr ℓ). This case does not apply as the sets of actuators of M and O are disjoint.
- $P_1 \parallel P_3 \xrightarrow{\tau} P'_1 \parallel P'_3$ by the application of on rule among (Res), (Rec), (Then) and (Else). This case does not apply to parallel processes.

- $\alpha = \text{deadlock}$. Suppose that $M \uplus O \xrightarrow{\text{deadlock}} M' \uplus O'$ is derived by an application of rule (Deadlock). This case is not admissible as the invariants of M , N and O are satisfied.
- $\alpha = \text{tick}$. Suppose that $M \uplus O \xrightarrow{\text{tick}} M' \uplus O'$ is derived by an application of rule (Time). This implies $P_1 \parallel P_3 \xrightarrow{\text{tick}} P'_1 \parallel P'_3$, for some P'_1 and P'_3 , $M' = E_1; S'_1 \bowtie P'_1$ and $O = E_3; S'_3 \bowtie P'_3$, with $S'_1 \in \text{next}(E_1; S_1)$ and $S'_3 \in \text{next}(E_3; S_3)$. As $P_1 \parallel P_3 \xrightarrow{\text{tick}} P'_1 \parallel P'_3$ can only be derived by an application of rule (TimePar), it follows that $P_1 \xrightarrow{\text{tick}} P'_1$ and $P_3 \xrightarrow{\text{tick}} P'_3$. Since the invariant of M is satisfied, by an application of rule (Time) we can derive $M \xrightarrow{\text{tick}} M'$ with $M' = E_1; S'_1 \bowtie P'_1$. As $M \sqsubseteq N$, there exists a derivation $N \xRightarrow{\hat{t}} N'' \xrightarrow{\text{tick}} N''' \xRightarrow{\hat{t}} N'$, for some $N' = E_2; S'_2 \bowtie P'_2$, $N'' = E_2; S''_2 \bowtie P''_2$, $N''' = E_2; S'''_2 \bowtie P'''_2$, with $S'''_2 \in \text{next}(E_2; S'_2)$. By several applications of rule (Par) we can derive that $N \uplus O \xRightarrow{\hat{t}} N'' \uplus O$ and $N''' \uplus O' \xRightarrow{\hat{t}} N' \uplus O'$. In order to conclude the proof, it is sufficient to prove $N'' \uplus O \xrightarrow{\text{tick}} N''' \uplus O'$. By the definition of rule (Time), from $N'' \xrightarrow{\text{tick}} N'''$ it follows that $P''_2 \xrightarrow{\text{tick}} P'''_2$. As $P_3 \xrightarrow{\text{tick}} P'_3$, by an application of rule (TimePar) it follows that $P''_2 \parallel P_3 \xrightarrow{\text{tick}} P'''_2 \parallel P'_3$. Since $S'''_2 \in \text{next}(E_2; S'_2)$ and $S'_3 \in \text{next}(E_3; S_3)$ we can derive that $S'''_2 \uplus S'_3 \in \text{next}(E_2; S'_2) \cup \text{next}(E_3; S_3)$. By an application of rule (Time) we have $N'' \uplus O \xrightarrow{\text{tick}} N''' \uplus O'$ and hence $N \uplus O \xRightarrow{\text{tick}} N' \uplus O'$, as required.
- $\alpha = \text{unsafe}$. Suppose that $M \uplus O \xrightarrow{\text{unsafe}} M' \uplus O'$ is derived by an application of rule (Safety). This is similar to the case $\alpha = \bar{c}v$ by considering the fact that $\xi_x \notin \text{safe}$ implies that $\xi_x \cup \xi_{x'} \notin \text{safe} \cup \text{safe}'$, for any $\xi_{x'}$ and any safe' .

Inductive case. We have to prove that $M \uplus O = M_0 \uplus O_0 \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n} M_n \uplus O_n$ implies $N \uplus O = N_0 \uplus O_0 \xRightarrow{\bar{\alpha}_1} \dots \xRightarrow{\bar{\alpha}_n} N_n \uplus O_n$. We can use the inductive hypothesis to easily deal with the first $n - 1$ actions and resort to the base case to handle the n^{th} action.

- (2) We have to prove that $M \sqsubseteq N$ implies $M \parallel P \sqsubseteq N \parallel P$, for any pure-logical process P . This is a special case of (1) as $M \parallel P = M \uplus (\emptyset; \emptyset \bowtie P)$ and $N \parallel P = N \uplus (\emptyset; \emptyset \bowtie P)$, where $\emptyset; \emptyset \bowtie P$ is a CPS with no physical process in it, only logics.
- (3) We have to prove that $M \sqsubseteq N$ implies $M \setminus c \sqsubseteq N \setminus c$, for any channel c . For any derivation $M \setminus c \xrightarrow{t} M' \setminus c$ we can easily derive that $M \xrightarrow{t} M'$ with c not occurring in t . Since $M \sqsubseteq N$, it follows that $N \xRightarrow{\hat{t}} N'$, for some N' . Since c does not appear in t , we can easily derive that $N \setminus c \xRightarrow{\hat{t}} N' \setminus c$, as required.

□

In order to prove Theorem 2 we adapt to CCPSA two standard lemmata used in process calculi theory to compose and decompose the actions performed by a compound system.

LEMMA 2 (DECOMPOSING SYSTEM ACTIONS). *Let M and N be two CPSs in CCPSA. Then,*

- if $M \uplus N \xrightarrow{\text{tick}} M' \uplus N'$, for some M' and N' , then $M \xrightarrow{\text{tick}} M'$ and $N \xrightarrow{\text{tick}} N'$;
- if $M \uplus N \xrightarrow{\text{deadlock}} M \uplus N$, then $M \xrightarrow{\text{deadlock}} M$ or $N \xrightarrow{\text{deadlock}} N$;
- if $M \uplus N \xrightarrow{\tau} M' \uplus N'$, for some M' and N' , due to a channel synchronisation between M and N , then either $M \xrightarrow{\bar{c}v} M'$ and $N \xrightarrow{cv} N'$, or $M \xrightarrow{cv} M'$ and $N \xrightarrow{\bar{c}v} N'$, for some channel c ;
- if $M \uplus N \xrightarrow{\alpha} M' \uplus N'$, for some M' and N' , $\alpha \neq \text{tick}$, not due to a channel synchronisation between M and N , then either $M \xrightarrow{\alpha} M$ and $N = N'$, or $N \xrightarrow{\alpha} N$ and $M = M'$.

LEMMA 3 (COMPOSING SYSTEM ACTIONS). *Let M and N be two CPSs of CCPSA. Then,*

- If $M \xrightarrow{\text{tick}} M'$ and $N \xrightarrow{\text{tick}} N'$, for some M' and N' , then $M \uplus N \xrightarrow{\text{tick}} M' \uplus N'$;
- If $N \xrightarrow{\text{deadlock}} N$ and $M \xrightarrow{\alpha} M'$, for some M' and $\alpha \neq \text{tick}$, then $M \uplus N \xrightarrow{\alpha} M' \uplus N$ and $N \uplus M \xrightarrow{\alpha} N \uplus M'$.

PROOF OF THEOREM 2. Here, we prove case (1) of the theorem. The proofs of cases (2) and (3) are similar to the corresponding ones of Theorem 1.

We prove that $M \sqsubseteq_{m..n} N$ implies that there are $m', n' \in \mathbb{N}^+ \cup \infty$, with $m'..n' \subseteq m..n$ such that $M \uplus O \sqsubseteq_{m'..n'} N \uplus O$. We prove separately that $m' \geq m$ and $n' \leq n$.

- $m' \geq m$. We recall that $m, m' \in \mathbb{N}^+$. If $m = 1$, then we trivially have $m' \geq 1 = m$. Otherwise, since m is the minimum integer for which there is a trace t , with $\#\text{tick}(t) = m - 1$, such that $M \xrightarrow{t}$ and $N \not\xrightarrow{t}$, then for any trace t , with $\#\text{tick}(t) < m - 1$ and such that $M \xrightarrow{t}$, it holds that $N \not\xrightarrow{t}$. As done in the proof of case (1) of Theorem 1, we can derive that for any trace t , with $\#\text{tick}(t) < m - 1$ and such that $M \uplus O \xrightarrow{t}$ it holds that $N \uplus O \not\xrightarrow{t}$. This implies the required condition, $m' \geq m$.
- $n' \leq n$. We recall that n is the infimum element of $\mathbb{N}^+ \cup \{\infty\}$, $n \geq m$, such that whenever $M \xrightarrow{t_1} M'$, with $\#\text{tick}(t_1) = n - 1$, there is t_2 , with $\#\text{tick}(t_1) = \#\text{tick}(t_2)$, such that $N \xrightarrow{t_2} N'$, for some N' , and $M' \sqsubseteq N'$. Now, if $M \uplus O \xrightarrow{t} M' \uplus O'$, with $\#\text{tick}(t) = n - 1$, by Lemma 2 we can split the trace t by extracting the actions performed by M and those performed by O . Thus, there exist two traces $M \xrightarrow{t_1} M'$ and $O \xrightarrow{t_3} O'$, with $\#\text{tick}(t_1) = \#\text{tick}(t_3) = n - 1$ whose combination has generated the trace $M \uplus O \xrightarrow{t} M' \uplus O'$. As $M \sqsubseteq_{m..n} N$, from $M \xrightarrow{t_1} M'$ we know that there is a trace t_2 , with $\#\text{tick}(t_1) = \#\text{tick}(t_2)$, such that $N \xrightarrow{t_2} N'$, for some N' , and $M' \sqsubseteq N'$. Since $N \xrightarrow{t_2} N'$ and $O \xrightarrow{t_3} O'$, by an application of Lemma 3 we can build a trace $N \uplus O \xrightarrow{t'} N' \uplus O'$, for some t' such that $\#\text{tick}(t) = \#\text{tick}(t') = n - 1$. As $M' \sqsubseteq N'$, by Theorem 1 we can derive that $M' \uplus O' \sqsubseteq N' \uplus O'$. This implies that $n' \leq n$.

□

In order to prove Theorem 3, we introduce the following lemma.

LEMMA 4. *Let M be an honest and sound CPS, C an arbitrary class of attacks, and A an attack of a class $C' \leq C$. Whenever $M \parallel A \xrightarrow{t} M' \parallel A'$, then $M \parallel \text{Top}(C) \xrightarrow{i} M' \parallel \prod_{i \in I} \text{Att}(i, \#\text{tick}(t) + 1, C(i))$.*

PROOF. Let us define $\text{Top}^h(C)$ as the attack process $\prod_{i \in I} \text{Att}(i, h, C(i))$. Then, $\text{Top}^1(C) = \text{Top}(C)$. The proof is by mathematical induction on the length k of the trace t .

Base case. $k = 1$. This means $t = \alpha$, for some action α . We proceed by case analysis on α .

- 1765 • $\alpha = \bar{c}v$. As the attacker A does not use communication channels, from $M \parallel A \xrightarrow{\bar{c}v} M' \parallel A'$
 1766 we can derive that $A = A'$ and $M \xrightarrow{\bar{c}v} M'$. Thus, by applications of rules (Par) and (Out) we
 1767 derive $M \parallel \text{Top}(C) \xrightarrow{\bar{c}v} M' \parallel \text{Top}^1(C) = M' \parallel \text{Top}(C)$.
 1768
 1769 • $\alpha = cv$. This case is similar to the previous one.
 1770
 1771 • $\alpha = \tau$. There are five sub-cases.
 1772 – Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule (SensRead). Since the attacker
 1773 A performs only malicious actions, from $M \parallel A \xrightarrow{\tau} M' \parallel A'$ we can derive that $A = A'$
 1774 and $P \xrightarrow{s?v} P'$ for some process P and P' such that $M = E; S \bowtie P$ and $M' = E; S \bowtie P'$. By
 1775 considering $\text{rnd}(\{\text{true}, \text{false}\}) = \text{false}$ for any process $\text{Att}(t, 1, C(t))$, we have that $\text{Top}(C)$
 1776 can only perform a tick action, and $\text{Top}(C) \xrightarrow{\text{tick}} \text{Top}^1(C)$. Hence, by an application of rules (Par)
 1777 and (SensRead) we derive $M \parallel \text{Top}(C) \xrightarrow{\tau} M' \parallel \text{Top}^1(C) = M' \parallel \text{Top}(C)$.
 1778 – Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule (ActWrite). This case is similar
 1779 to the previous one.
 1780 – Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule ($\not\{SensSniff\}$). Since M
 1781 is sound it follows that $M = M'$ and $A \xrightarrow{\not\{s?v\}} A'$. This entails $1 \in C'(\not\{s?\}) \subseteq C(\not\{s?\})$.
 1782 By assuming $\text{rnd}(\{\text{true}, \text{false}\}) = \text{true}$ for the process $\text{Att}(\not\{s?, 1, C(\not\{s?\})$, it follows that
 1783 $\text{Top}(C) \xrightarrow{\not\{s?v\}} \text{Top}^1(C) = \text{Top}(C)$. Hence, by applying the rules (Par) and ($\not\{SensRead\}$) we
 1784 derive $M \parallel \text{Top}(C) \xrightarrow{\tau} M' \parallel \text{Top}^1(C) = M' \parallel \text{Top}(C)$.
 1785
 1786 – Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule ($\not\{ActIntgr\}$). Since M is
 1787 sound it follows that $M = M'$ and $A \xrightarrow{\not\{a!v\}} A'$. As a consequence, $1 \in C'(\not\{a!\}) \subseteq C(\not\{a!\})$.
 1788 By assuming $\text{rnd}(\{\text{true}, \text{false}\}) = \text{true}$ and $\text{rnd}(\mathbb{R}) = v$ for the process $\text{Att}(\not\{a!, 1, C(\not\{a!\})$,
 1789 it follows that $\text{Top}(C) \xrightarrow{\not\{a!v\}} \text{Top}^1(C) = \text{Top}(C)$. Thus, by applying the rules (Par) and
 1790 ($\not\{ActIntgr\}$) we derive $M \parallel \text{Top}(C) \xrightarrow{\tau} M' \parallel \text{Top}^1(C) = M' \parallel \text{Top}(C)$.
 1791
 1792 – Let $M \parallel A \xrightarrow{\tau} M' \parallel A'$ be derived by an application of rule (Tau). Let $M = E; S \bowtie P$ and
 1793 $M' = E'; S \bowtie P'$. First, we consider the case when $P \parallel A \xrightarrow{\tau} P' \parallel A'$ is derived by an
 1794 application of either rule ($\not\{SensIntgr\}$) or rule ($\not\{ActDrop\}$). Since M is sound and A can
 1795 perform only malicious actions, we have that: (i) either $P \xrightarrow{s?v} P'$ and $A \xrightarrow{\not\{s!v\}} A'$ (ii) or
 1796 $P \xrightarrow{a!v} P'$ and $A \xrightarrow{\not\{a?v\}} A'$. We focus on the first case as the second one is similar.
 1797 Since $A \xrightarrow{\not\{s!v\}} A'$, we derive $1 \in C'(\not\{s!\}) \subseteq C(\not\{s!\})$, and $\text{Top}(C) \xrightarrow{\not\{s!v\}} \text{Top}^1(C) = \text{Top}(C)$,
 1798 by assuming $\text{rnd}(\{\text{true}, \text{false}\}) = \text{true}$ and $\text{rnd}(\mathbb{R}) = v$ for the process $\text{Att}(\not\{s!, 1, C(\not\{s!\})$.
 1799 Thus, by applying the rules ($\not\{SensIntgr\}$) and (Tau) we derive $M \parallel \text{Top}(C) \xrightarrow{\tau} M' \parallel$
 1800 $\text{Top}^1(C) = M' \parallel \text{Top}(C)$, as required.
 1801
 1802 To conclude the proof we observe that if $P \parallel A \xrightarrow{\tau} P' \parallel A'$ is derived by an application
 1803 of a rule different from ($\not\{SensIntgr\}$) and ($\not\{ActDrop\}$), then by inspection of Table 1 and
 1804 by definition of attacker, it follows that A can't perform a τ -action since A does not use
 1805 channel communication and performs only malicious actions. Thus, the only possibility is
 1806 that the τ -action is performed by P in isolation. As a consequence, by applying the rules
 1807 (Par) and (Tau), we derive $M \parallel \text{Top}(C) \xrightarrow{\tau} M' \parallel \text{Top}^1(C) = M' \parallel \text{Top}(C)$.
 1808
 1809 • $\alpha = \text{tick}$. In this case the transition $M \parallel A \xrightarrow{\text{tick}} M' \parallel A'$ is derived by an application of rule
 1810 (Time) because $M \xrightarrow{\text{tick}} M'$ and $A \xrightarrow{\text{tick}} A'$. Hence, it suffices to prove that $\text{Top}(C) \xrightarrow{\text{tick}}$
 1811 $\text{Top}^2(C)$. We consider two cases: $1 \in C(t)$ and $1 \notin C(t)$. If $1 \in C(t)$, then the transition
 1812
 1813

1814 $Att(i, 1, C(i)) \xrightarrow{\text{tick}} Att(i, 2, C(i))$ can be derived by assuming $rnd(\{\text{true}, \text{false}\}) = \text{false}$.
 1815 Moreover, since $rnd(\{\text{true}, \text{false}\}) = \text{false}$ the process $Att(i, 1, C(i))$ can only perform a tick
 1816 action. If $1 \notin C(i)$, then the process $Att(i, 1, C(i))$ can only perform a tick action. As a
 1817 consequence, $Att(i, 1, C(i)) \xrightarrow{\text{tick}} Att(i, 2, C(i))$ and $Top(C) \xrightarrow{\text{tick}} Top^2(C)$. By an application
 1818 of rule (Time), we derive $M \parallel Top(C) \xrightarrow{\text{tick}} M' \parallel Top^2(C)$.

- 1820 • $\alpha = \text{deadlock}$. This case is not admissible because $M \parallel A \xrightarrow{\text{deadlock}} M' \parallel A'$ would entail
 1821 $M \xrightarrow{\text{deadlock}} M'$. However, M is sound and it can't deadlock.
- 1822 • $\alpha = \text{unsafe}$. Again, this case is not admissible because M is sound.

1824 *Inductive case ($k > 1$).* We have to prove that $M \parallel A \xrightarrow{t} M' \parallel A'$ implies $M \parallel Top(C) \xrightarrow{\hat{t}} M' \parallel$
 1825 $Top^{\#\text{tick}(t)+1}(C)$. Since the length of t is greater than 1, it follows that $t = t'\alpha$, for some trace t'
 1826 and some action α . Thus, there exist M'' and A'' such that $M \parallel A \xrightarrow{t'} M'' \parallel A'' \xrightarrow{\alpha} M' \parallel A'$.
 1827 By inductive hypothesis, it follows that $M \parallel Top(C) \xrightarrow{\hat{t}'} M'' \parallel Top^{\#\text{tick}(t')+1}(C)$. To conclude
 1828 the proof, it is enough to show that $M'' \parallel A'' \xrightarrow{\alpha} M' \parallel A'$ implies $M'' \parallel Top^{\#\text{tick}(t')+1}(C) \xrightarrow{\hat{\alpha}}$
 1829 $M' \parallel Top^{\#\text{tick}(t)+1}(C)$. The reasoning is similar to that followed in the base case, except for actions
 1830 $\alpha = \text{deadlock}$ and $\alpha = \text{unsafe}$ that need to be treated separately. We prove the case $\alpha = \text{deadlock}$
 1831 as the case $\alpha = \text{unsafe}$ is similar.

1833 Let $M = E; S \bowtie P$. The transition $M'' \parallel A \xrightarrow{\text{deadlock}} M' \parallel A'$ must be derived by an application
 1834 of rule (Deadlock). This implies that $M'' = M'$, $A'' = A'$ and the state function of M is not in the
 1835 invariant set inv . Thus, by an application of rule (Deadlock) we derive

$$1836 M'' \parallel Top^{\#\text{tick}(t')+1}(C) \xrightarrow{\text{deadlock}} M' \parallel Top^{\#\text{tick}(t')+1}(C).$$

1838 Since $\#\text{tick}(t) + 1 = \#\text{tick}(t') + \#\text{tick}(\text{deadlock}) + 1 = \#\text{tick}(t') + 1$, it follows, as required, that

$$1840 M'' \parallel Top^{\#\text{tick}(t')+1}(C) \xrightarrow{\text{deadlock}} M' \parallel Top^{\#\text{tick}(t)+1}(C).$$

1841 □

1842 Everything is finally in place to prove Theorem 3.

1844 **PROOF OF THEOREM 3.** We have to prove that either $M \parallel A \sqsubseteq M$ or $M \parallel A \sqsubseteq_{m_2..n_2} M$, for some
 1845 m_2 and n_2 such that $m_2..n_2 \subseteq m_1..n_1$ ($m_2 = 1$ and $n_2 = \infty$ if the two systems are completely
 1846 unrelated). The proof proceeds by contradiction. Suppose that $M \parallel A \not\sqsubseteq M$ and $M \parallel A \not\sqsubseteq_{m_2..n_2} M$,
 1847 with $m_2..n_2 \not\subseteq m_1..n_1$. We distinguish two cases: either $n_1 = \infty$ or $n_1 \in \mathbb{N}^+$.

1848 If $n_1 = \infty$, then it must be $m_2 < m_1$. Since $M \parallel A \sqsubseteq_{m_2..n_2} M$, by Definition 10 there is a trace t ,
 1849 with $\#\text{tick}(t) = m_2 - 1$, such that $M \parallel A \xrightarrow{t}$ and $M \not\xrightarrow{\hat{t}}$. By Lemma 4, this entails $M \parallel Top(C) \xrightarrow{\hat{t}}$.
 1850 Since $M \not\xrightarrow{\hat{t}}$ and $\#\text{tick}(t) = m_2 - 1 < m_2 < m_1$, this contradicts $M \parallel Top(C) \sqsubseteq_{m_1..n_1} M$.

1852 If $n_1 \in \mathbb{N}^+$, then $m_2 < m_1$ and/or $n_1 < n_2$, and we reason as in the previous case. □

1854 **PROOF OF THEOREM 4.** We consider the two parts of the statement separately.

1855 *Definitive impact.* By an application of Lemma 4 we have that $M \parallel A \xrightarrow{t}$ entails $M \parallel Top(C) \xrightarrow{\hat{t}}$.
 1856 This implies $M \parallel A \sqsubseteq M \parallel Top(C)$. Thus, if $M \parallel Top(C) \sqsubseteq M[\xi_w \leftarrow \xi_w + \xi]$, for $\xi \in \mathbb{R}^X$, $\xi > 0$, then,
 1857 by transitivity of \sqsubseteq , it follows that $M \parallel A \sqsubseteq M[\xi_w \leftarrow \xi_w + \xi]$.

1858 *Pointwise impact.* The proof proceeds by contradiction. Suppose $\xi' > \xi$. Since $Top(C)$ has a
 1859 pointwise impact ξ at time m , it follows that ξ is given by:

$$1861 \inf \{ \xi'' : \xi'' \in \mathbb{R}^X \wedge M \parallel Top(C) \sqsubseteq_{m..n} M[\xi_w \leftarrow \xi_w + \xi''] \}, n \in \mathbb{N}^+ \cup \infty \}.$$

1862

Similarly, since A has a pointwise impact ξ' at time m' , it follows that ξ' is given by

$$\inf \left\{ \xi'' : \xi'' \in \mathbb{R}^{\hat{X}} \wedge M \parallel A \sqsubseteq_{m'..n} M[\xi_w \leftarrow \xi_w + \xi''], n \in \mathbb{N}^+ \cup \infty \right\}.$$

Now, if $m = m'$, then $\xi \geq \xi'$ because $M \parallel A \xrightarrow{t}$ entails $M \parallel \text{Top}(C) \xrightarrow{\hat{t}}$ due to an application of Lemma 4. This is contradiction with the fact that $\xi < \xi'$. Thus, it must be $m' < m$. Now, since both ξ and ξ' are the infimum functions and since $\xi' > \xi$, there are $\bar{\xi}$ and $\bar{\xi}'$, with $\xi \leq \bar{\xi} \leq \xi' \leq \bar{\xi}'$ such that: (i) $M \parallel \text{Top}(C) \sqsubseteq_{m..n} M[\xi_w \leftarrow \xi_w + \bar{\xi}]$, for some n ; (ii) $M \parallel A \sqsubseteq_{m'..n'} M[\xi_w \leftarrow \xi_w + \bar{\xi}']$, for some n' .

From $M \parallel A \sqsubseteq_{m'..n'} M[\xi_w \leftarrow \xi_w + \bar{\xi}']$ it follows that there exists a trace t with $\#\text{tick}(t) = m' - 1$ such that $M \parallel A \xrightarrow{t}$ and $M[\xi_w \leftarrow \xi_w + \bar{\xi}'] \not\xrightarrow{\hat{t}}$. Since $\bar{\xi} \leq \bar{\xi}'$, by monotonicity (Proposition 6), we deduce that $M[\xi_w \leftarrow \xi_w + \bar{\xi}] \not\xrightarrow{\hat{t}}$. Moreover, by Lemma 4 $M \parallel A \xrightarrow{t}$ entails $M \parallel \text{Top}(C) \xrightarrow{\hat{t}}$.

Summarising, there exists a trace t' with $\#\text{tick}(t') = m' - 1$ such that $M \parallel \text{Top}(C) \xrightarrow{t'}$ and $M[\xi_w \leftarrow \xi_w + \bar{\xi}] \not\xrightarrow{\hat{t}'}$. However, this, together with $m' < m$, is in contradiction with the fact (i) above saying that $M \parallel \text{Top}(C) \sqsubseteq_{m..n} M[\xi_w \leftarrow \xi_w + \bar{\xi}]$, for some n . As a consequence it must be $\xi' \leq \xi$ and $m' \leq m$. This concludes the proof. \square

Received XYZ 2019; revised XYZ; accepted XYZ