2017 IEEE 30th Computer Security Foundations Symposium

A Formal Approach to Cyber-Physical Attacks

Ruggero Lanotte
Dipartimento di Scienza e Alta Tecnologia
Università dell'Insubria, Como, Italy

Massimo Merro and Riccardo Muradore
Dipartimento di Informatica
Università degli Studi di Verona, Italy

Luca Viganò
Department of Informatics
King's College London, UK

***Abstract*—We apply formal methods to lay and streamline theoretical foundations to reason about Cyber-Physical Systems (CPSs) and cyber-physical attacks. We focus on integrity and DoS attacks to sensors and actuators of CPSs, and on the timing aspects of these attacks. Our contributions are threefold: (1) we define a hybrid process calculus to model both CPSs and cyber-physical attacks; (2) we define a threat model of cyber-physical attacks and 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. We illustrate definitions and results by means of a non-trivial engineering application.**

***Index Terms*—Cyber-physical system security, formal methods, theoretical foundation, process calculus.**

I. Introduction

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 processing components* are connected to control stations. Different kinds of CPSs include *supervisory control and data acquisition (SCADA)*, *programmable logic controllers (PLC)* and distributed control systems.

In recent years there has been a dramatic increase in the number of attacks to the security of cyber-physical and critical systems, e.g., manipulating sensor readings and, in general, influencing physical processes to bring the system into a state desired by the attacker. Many (in)famous examples have been so impressive to make the international news, e.g., the Stuxnet worm, which reprogrammed PLCs of nuclear centrifuges in Iran [6] or the attack on a sewage treatment facility in Queensland, Australia, which manipulated the SCADA system to release raw sewage into local rivers and parks [25].

As stated in [10], 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. The works that have taken up these challenges range from proposals of different notions of cyber-physical security and attacks (e.g., [3], [10], [13], to name a few) to pioneering extensions to CPS security of standard formal approaches (e.g., [3], [4], [29]). However, to the best of our knowledge, a systematic formal approach to cyber-physical attacks is still to be fully developed.

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.

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 statically detect attacks to physical devices of CPSs. A straightforward utilisation of these methodologies is for *model-checking*, in order to be able to statically analyse security properties of CPSs before their practical implementation and deployment. 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).

Our contributions are threefold. The first contribution is the definition of a *hybrid process calculus*, called `CCPSA`, to formally specify both CPSs and cyber-physical 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.

[1] See [31] for a taxonomy of the time-scale models used to represent CPSs.

DOI 10.1109/CSF.2017.12

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* [11] 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 trace-based system preorders: a *trace preorder*, $\sqsubseteq$, and a *timed variant*, $\sqsubseteq_{m..n}$, for $m, n \in \mathbb{N}^+ \cup \{\infty\}$, which takes into account discrepancies of execution traces within the 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 attacks that can manipulate sensor and/or actuator signals in order to drive a CPS into an undesired state [26]. Cyber-physical attacks typically tamper with both the physical (sensors and actuators) and the cyber layer. In our threat model, communication cannot be manipulated by the attacker, who instead may compromise (unsecured) physical devices, which is our focus. 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 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 $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$. 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. Thus, a class C is a total function that associates to any malicious use (read/write) of any physical device (sensor/actuator) a possibly empty set of time instants in which the attacker tampers with that specific device.

As observed in [13], timing is a critical issue in CPSs because the physical state of a system changes continuously over time and, as the system evolves in time, some states might be more vulnerable to attacks than others. 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 [14]. 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 [27], hours to heat a tank of water or burn out a motor, and days to destroy centrifuges [6].

In order to make security assessments on our CPSs, we adopt a well-known approach called *Generalized Non Deducibility on Composition (GNDC)* [7]. Thus, in CCPSA, we say that a

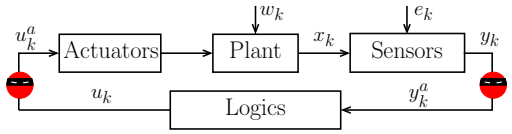

Fig. 1. Our threat model for CPSs

CPS Sys *tolerates* a cyber-physical attack A if

$$Sys \parallel A \sqsubseteq Sys \ .$$

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

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

$$Sys \parallel A \sqsubseteq_{m'..n'} Sys \ .$$

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, $Top(C)$, and prove that if a CPS tolerates $Top(C)$ then it tolerates all attacks A of class C. Similarly, if a CPS is vulnerable to $Top(C)$, in the time interval $m'..n'$, then no attacks of class C 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* and investigate possible *quantifications of the chances* for an attack of being successful when attacking a CPS. This is important since, in industrial CPSs, before taking any countermeasure against an attack, engineers typically first try to estimate the impact of the attack on the system functioning (e.g., performance and security) and weigh it against the cost of stopping the plant. If this cost is higher than the damage caused by the attack (as is sometimes the case), then engineers might actually decide to let the system continue its activities even under attack. We thus provide a *metric* to estimate the deviation of the system under attack with respect to expected behaviour, according to its evolution law and the uncertainty of the model. Then, we prove that the impact of the most powerful attack $Top(C)$ represents an upper bound for the impact of any attack A of class C.

We introduce a non-trivial *running example* taken from an engineering application and use it to illustrate our definitions and cases of CPSs that tolerate certain attacks, and of CPSs that suffer from attacks that drag them towards undesired behaviours. We remark that while we have kept the example simple, it is actually far from trivial and designed to describe a wide number of attacks, as will become clear below.

All the results presented in the paper have been formally proven; due to lack of space, proofs have been sketched or omitted and full details can be found in [16]. Moreover, the

behaviour of our running example and of most of the cyber-physical attacks appearing in the paper have been simulated in MATLAB.

Organisation: In § II, we give syntax and semantics of CCPSA. In § III, we define cyber-physical attacks and provide sufficient criteria for attack tolerance/vulnerability. In § IV, we estimate the impact of attacks on CPSs, and investigate possible quantifications of the success chances of an attack. In § V, we draw conclusions and discuss related and future work.

II. 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 paper [15], with specific features to formalise and study attacks to physical devices. Let us start with some preliminary notations.

Notation 1. *We use $x, x_k \in \mathcal{X}$ for* state variables*, $c, d \in \mathcal{C}$ for* communication channels*, $a, a_k \in \mathcal{A}$ for* actuator devices*, $s, s_k \in \mathcal{S}$ for* sensors devices*, and p, q for both sensors and actuators (generically called* physical devices*).* Values*, ranged over by $v, v' \in \mathcal{V}$, are built from basic values, such as Booleans, integers and real numbers.* State variables *are metavariables for physical properties like $temperature$, $pressure$, etc.* Actuator names *are metavariables for actuator devices like $valve$, $light$, etc. Similarly,* sensor names *are metavariables for sensor devices, e.g., a sensor $thermometer$.*

Given a set of names $\mathcal{N}$, we write $\mathbb{R}^{\mathcal{N}}$ to denote the set of functions $[\mathcal{N} \to \mathbb{R}]$ assigning a real to each name in $\mathcal{N}$. For $\xi \in \mathbb{R}^{\mathcal{N}}$, $n \in \mathcal{N}$ and $v \in \mathbb{R}$, we write $\xi[n \mapsto v]$ for the function $\psi \in \mathbb{R}^{\mathcal{N}}$ such that $\psi(m) = \xi(m)$, for any $m \neq n$, and $\psi(n) = v$.

Finally, we distinguish between real intervals*, such as $(m, n]$, for $m \in \mathbb{R}$ and $n \in \mathbb{R} \cup \{\infty\}$, and* integer intervals*, written $m..n$, for $m \in \mathbb{N}$ and $n \in \mathbb{N} \cup \{\infty\}$. As we will adopt a discrete notion of time, we will use integer intervals to denote* time intervals*.*

Definition 1 (Cyber-physical system)**.** *In CCPSA, a* cyber-physical system *consists of two components:*

- *a* physical environment *E that encloses all physical aspects of a system (state variables, physical devices, evolution law, etc.) and*
- *a* cyber component *P that interacts with sensors and actuators of the system, and can communicate, via channels, with other cyber components of the same or of other CPSs.*

Given a set $\mathcal{S}$ of secured physical devices *of E, we write $E \bowtie_{\mathcal{S}} P$ to denote the resulting CPS, and use M and N to range over CPSs. We write $E \bowtie P$ when $\mathcal{S} = \emptyset$.*

In a CPS $E \bowtie_{\mathcal{S}} P$, the "secured" devices in $\mathcal{S}$ are accessed in a protected way and hence they cannot be attacked.[2]

[2]The presence of battery-powered devices interconnected through wireless networks prevents the en-/decryption of all packets due to energy constraints.

Let us now define physical environments E and cyber components P in order to formalise our proposal for modelling (and reasoning about) CPSs and cyber-physical attacks.

Definition 2 (Physical environment)**.** *Let $\hat{\mathcal{X}} \subseteq \mathcal{X}$ be a set of state variables, $\hat{\mathcal{A}} \subseteq \mathcal{A}$ be a set of actuators, and $\hat{\mathcal{S}} \subseteq \mathcal{S}$ be a set of sensors. A* physical environment *E is an 8-tuple $\langle \xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, evol, \xi_{\mathrm{e}}, meas, inv, safe \rangle$, where:*

- *$\xi_{\mathrm{x}} \in \mathbb{R}^{\hat{\mathcal{X}}}$ is the* state function*,*
- *$\xi_{\mathrm{u}} \in \mathbb{R}^{\hat{\mathcal{A}}}$ is the* actuator function*,*
- *$\xi_{\mathrm{w}} \in \mathbb{R}^{\hat{\mathcal{X}}}$ is the* uncertainty function*,*
- *$evol : \mathbb{R}^{\hat{\mathcal{X}}} \times \mathbb{R}^{\hat{\mathcal{A}}} \times \mathbb{R}^{\hat{\mathcal{X}}} \to 2^{\mathbb{R}^{\hat{\mathcal{X}}}}$ is the* evolution map*,*
- *$\xi_{\mathrm{e}} \in \mathbb{R}^{\hat{\mathcal{S}}}$ is the* sensor-error function*,*
- *$meas : \mathbb{R}^{\hat{\mathcal{X}}} \times \mathbb{R}^{\hat{\mathcal{S}}} \to 2^{\mathbb{R}^{\hat{\mathcal{S}}}}$ is the* measurement map*,*
- *$inv : \mathbb{R}^{\hat{\mathcal{X}}} \to \{\mathsf{true}, \mathsf{false}\}$ is the* invariant function*,*
- *$safe : \mathbb{R}^{\hat{\mathcal{X}}} \to \{\mathsf{true}, \mathsf{false}\}$ is the* safety function*.*

All the functions defining an environment are total functions*.*

The *state function* ξ_{x} returns the current value (in $\mathbb{R}$) associated to each state variable of the system. The *actuator function* ξ_{u} returns the current value associated to each actuator. The *uncertainty function* ξ_{w} returns the uncertainty associated to each state variable. Thus, given a state variable $x \in \hat{\mathcal{X}}$, $\xi_{\mathrm{w}}(x)$ returns the maximum distance between the real value of x and its representation in the model. Later in the paper, we will be interested in comparing the accuracy of two systems. Thus, for $\xi_{\mathrm{w}}, \xi'_{\mathrm{w}} \in \mathbb{R}^{\hat{\mathcal{X}}}$, we will write $\xi_{\mathrm{w}} \leq \xi'_{\mathrm{w}}$ if $\xi_{\mathrm{w}}(x) \leq \xi'_{\mathrm{w}}(x)$, for any $x \in \hat{\mathcal{X}}$. Similarly, we write $\xi_{\mathrm{w}} + \xi'_{\mathrm{w}}$ to denote the function $\xi''_{\mathrm{w}} \in \mathbb{R}^{\hat{\mathcal{X}}}$ such that $\xi''_{\mathrm{w}}(x) = \xi_{\mathrm{w}}(x) + \xi'_{\mathrm{w}}(x)$, for any $x \in \hat{\mathcal{X}}$.

Given a state function, an actuator function, and an uncertainty function, the *evolution map* $evol$ returns the set of next *admissible state functions*. It models the *evolution law* of the physical system, where changes made on actuators may reflect on state variables. Since we assume an uncertainty in our models, $evol$ does not return a single state function but a set of possible state functions. $evol$ is obviously *monotone* with respect to uncertainty: if $\xi_{\mathrm{w}} \leq \xi'_{\mathrm{w}}$ then $evol(\xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}) \subseteq evol(\xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi'_{\mathrm{w}})$.

Both the state function and the actuator function are supposed to change during the evolution of the system, whereas the uncertainty function is constant.[3]

The *sensor-error function* ξ_{e} returns the maximum error associated to each sensor in $\hat{\mathcal{S}}$. Again due to the presence of the sensor-error function, the *measurement map* $meas$, given the current state function, returns a set of admissible measurement functions rather than a single one.

The *invariant function* inv represents the conditions that the state variables must satisfy to allow for the evolution of the system. A CPS whose state variables don't satisfy the invariant is in *deadlock*.

The *safety function* $safe$ represents the conditions that the state variables must satisfy to consider the CPS in a safe state.

[3]Note that, although the uncertainty function is constant, it can be used in the evolution map in an arbitrary way (e.g., it could have a heavier weight when a state variable reaches extreme values). Another possibility is to model the uncertainty function by means of a probability distribution.

Intuitively, if a CPS gets in an unsafe state, then its functionality may get compromised.

In the following, we use a specific notation for the replacement of a single component of an environment with a new one of the same kind; for instance, for $E = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, evol, \xi_{\mathrm{e}}, meas, inv, safe \rangle$, we write $E[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}]$ to denote $\langle \xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi'_{\mathrm{w}}, evol, \xi_{\mathrm{e}}, meas, inv, safe \rangle$.

Let us now introduce a *running example* to illustrate our approach. We remark that while we have kept the example simple, it is actually far from trivial and designed to describe a wide number of attacks. A more complex example (say, with n sensors and m actuators) wouldn't have been more instructive but just made the paper more dense.

Example 1 (Physical environment of the CPS Sys)**.** *Consider a CPS Sys in which the temperature of an engine is maintained within a specific range by means of a cooling system. The physical environment Env of Sys is constituted by: (i) a state variable $temp$ containing the current temperature of the engine, and a 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) an actuator $cool$ to turn on/off the cooling system; (iii) a sensor s_{t} (such as a thermometer or a thermocouple) measuring the temperature of the engine; (iv) an uncertainty $\delta = 0.4$ associated to the only variable $temp$; (v) the evolution law for the two state variables: the variable $temp$ is increased (resp., is decreased) of one degree per time unit if the cooling system is inactive (resp., active), whereas the variable $stress$ contains an integer that is increased each time the current temperature is above 9.9 degrees, and dropped to 0 otherwise; (vi) an error $\epsilon = 0.1$ associated to the only sensor s_{t}; (vii) a measurement map to get the values detected by sensor s_{t}, up to its error ϵ; (viii) an invariant function saying that the system gets faulty when the temperature of the engine gets out of the range $[0, 50]$; (ix) a safety function to say that the system moves to an unsafe state when the level of stress reaches the threshold 5.*

Formally, $Env = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, evol, \xi_{\mathrm{e}}, meas, inv \rangle$ with:

- $\xi_{\mathrm{x}} \in \mathbb{R}^{\{temp, stress\}}$ *and* $\xi_{\mathrm{x}}(temp) = 0$ *and* $\xi_{\mathrm{x}}(stress){=}0$*;*
- $\xi_{\mathrm{u}} \in \mathbb{R}^{\{cool\}}$ *and* $\xi_{\mathrm{u}}(cool) = \mathsf{off}$*; for the sake of simplicity, we can assume ξ_{u} to be a mapping $\{cool\} \to \{\mathsf{on}, \mathsf{off}\}$ such that $\xi_{\mathrm{u}}(cool) = \mathsf{off}$ if $\xi_{\mathrm{u}}(cool) \geq 0$, and $\xi_{\mathrm{u}}(cool) = \mathsf{on}$ if $\xi_{\mathrm{u}}(cool) < 0$;*
- $\xi_{\mathrm{w}} {\in} \mathbb{R}^{\{temp, stress\}}$*,* $\xi_{\mathrm{w}}(temp){=}0.4{=}\delta$ *and* $\xi_{\mathrm{w}}(stress){=}0$*;*
- $evol(\xi^i_{\mathrm{x}}, \xi^i_{\mathrm{u}}, \xi_{\mathrm{w}})$ *is the set of* $\xi \in \mathbb{R}^{\{temp, stress\}}$ *such that:*
  - $\xi(temp) = \xi^i_{\mathrm{x}}(temp) + heat(\xi^i_{\mathrm{u}}, cool) + \gamma$*, with* $\gamma \in [-\delta, +\delta]$ *and* $heat(\xi^i_{\mathrm{u}}, cool) = -1$ *if* $\xi^i_{\mathrm{u}}(cool) = \mathsf{on}$ *(active cooling), and* $heat(\xi^i_{\mathrm{u}}, cool) = +1$ *if* $\xi^i_{\mathrm{u}}(cool) = \mathsf{off}$ *(inactive cooling);*
  - $\xi(stress) = \min(5\,,\, \xi^i_{\mathrm{x}}(stress){+}1)$ *if* $\xi^i_{\mathrm{x}}(temp) > 9.9$*;* $\xi(stress) = 0$*, otherwise;*
- $\xi_{\mathrm{e}} \in \mathbb{R}^{\{s_{\mathrm{t}}\}}$ *and* $\xi_{\mathrm{e}}(s_{\mathrm{t}}) = 0.1 = \epsilon$*;*
- $meas(\xi^i_{\mathrm{x}}, \xi_{\mathrm{e}}){=}\big\{\xi : \xi(s_{\mathrm{t}}){\in}[\xi^i_{\mathrm{x}}(temp){-}\epsilon\,,\, \xi^i_{\mathrm{x}}(temp){+}\epsilon]\big\}$*;*
- $inv(\xi_{\mathrm{x}}) = \mathsf{true}$ *if* $0 \leq \xi_{\mathrm{x}}(temp) \leq 50$*;* $inv(\xi_{\mathrm{x}}) = \mathsf{false}$*, otherwise.*
- $safe(\xi_{\mathrm{x}}) = \mathsf{true}$ *if* $\xi_{\mathrm{x}}(stress) < 5$*;* $safe(\xi_{\mathrm{x}}) = \mathsf{false}$*, if* $\xi_{\mathrm{x}}(stress) \geq 5$ *(the maximum value for stress is 5).*

Let us now formalise the cyber component of CPSs in CCPSA. Basically, we extend the *timed process algebra TPL* of [11] with two main ingredients:

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

The remaining constructs are the same as those of TPL.

Definition 3 (Processes)**.** Processes *are defined as follows:*

$$
\begin{array}{rcl}
P, Q & ::= & \mathsf{nil} \;\mid\; \mathsf{idle}.P \;\mid\; P \parallel Q \;\mid\; \mathsf{timeout}\lfloor \pi.P \rfloor Q \;\mid\; \\
 & & \mathsf{if}\,(b)\,\{P\}\,\mathsf{else}\,\{Q\} \;\mid\; P\backslash c \;\mid\; H\langle \tilde{w} \rangle \\
\pi & ::= & \mathsf{snd}\, c\langle v \rangle \;\mid\; \mathsf{rcv}\, c(x) \;\mid\; \mathsf{read}\, s(x) \;\mid\; \\
 & & \mathsf{write}\, a\langle v \rangle \;\mid\; \mathsf{read}\, \lightning p(x) \;\mid\; \mathsf{write}\, \lightning p\langle v \rangle \;\;.
\end{array}
$$

We write nil for the *terminated process*. The process $\mathsf{idle}.P$ sleeps for one time unit and then continues as P. We write $P \parallel Q$ to denote the *parallel composition* of concurrent *threads*P and Q. The process $\mathsf{timeout}\lfloor \pi.P \rfloor Q$ denotes *prefixing with timeout*. Thus, $\mathsf{timeout}\lfloor \mathsf{snd}\, c\langle v \rangle.P \rfloor Q$ sends the value v on channel c and, after that, it continues as P; otherwise, after one time unit, it evolves into Q. The process $\mathsf{timeout}\lfloor \mathsf{rcv}\, c(x).P \rfloor Q$ is the obvious counterpart for reception. The process $\mathsf{timeout}\lfloor \mathsf{read}\, s(x).P \rfloor Q$ reads the values detected by the sensor s, whereas $\mathsf{timeout}\lfloor \mathsf{write}\, a\langle v \rangle.P \rfloor Q$ writes on the actuator a. For $\pi \in \{\mathsf{read}\, \lightning p(x), \mathsf{write}\, \lightning p\langle v \rangle\}$, the process $\mathsf{timeout}\lfloor \pi.P \rfloor Q$ denotes the reading and the writing, respectively, of the physical device p (sensor or actuator) made by the *attack*. Thus, in CCPSA, *attack processes* have specific constructs to interact with physical devices.

The process $P\backslash c$ is the channel restriction operator of CCS. It is quantified over the set $\mathcal{C}$ of communication channels but we sometimes write $P\backslash\{c_1, c_2, \ldots, c_n\}$ to mean $P\backslash c_1\backslash c_2 \cdots \backslash c_n$. The process $\mathsf{if}\,(b)\,\{P\}\,\mathsf{else}\,\{Q\}$ is the standard conditional, where b is a decidable guard. In processes of the form $\mathsf{idle}.Q$ and $\mathsf{timeout}\lfloor \pi.P \rfloor Q$, the occurrence of Q is said to be *time-guarded*. The process $H\langle \tilde{w} \rangle$ denotes (guarded) recursion.

We assume a set of *process identifiers* ranged over by H, H_1, H_2. We write $H\langle w_1, \ldots, w_k \rangle$ to denote a recursive process H defined via an equation $H(x_1, \ldots, x_k) = P$, where (i) the tuple $x_1, \ldots, x_k$ 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 two constructs $\mathsf{timeout}\lfloor \mathsf{rcv}\, c(x).P \rfloor Q$ and $\mathsf{timeout}\lfloor \mathsf{read}\, \mu(x).P \rfloor Q$, with $\mu \in \{p, \lightning p\}$, 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.

Note that in CCPSA, a processes might use sensors and/or actuators which are not defined in the environment. To rule out ill-formed CPSs, we use the following definition.

Definition 4 (Well-formedness)**.** *Given a process P and an environment $E = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, evol, \xi_{\mathrm{e}}, meas, inv, safe \rangle$, the CPS $E \bowtie P$ is* well-formed *if: (i) for any sensor s mentioned in P, the function ξ_{e} is defined on s; (ii) for any actuator a mentioned in P, the function ξ_{u} is defined on a.*

Hereafter, we will always work with well-formed CPSs.

Notation 2. *To model* time-persistent prefixing*, we write $\pi.P$ for the process defined via the equation $Rcv = \mathsf{timeout}\lfloor \pi.P \rfloor Rcv$, where Rcv does not occur in P. Further, we write*

- $\mathsf{timeout}\lfloor \pi \rfloor Q$ *as an abbreviation for* $\mathsf{timeout}\lfloor \pi.\mathsf{nil} \rfloor Q$,
- $\mathsf{timeout}\lfloor \pi.P \rfloor$ *as an abbreviation for* $\mathsf{timeout}\lfloor \pi.P \rfloor \mathsf{nil}$,
- $\mathsf{snd}\, c$ *and* $\mathsf{rcv}\, c$*, when channel c is used for pure synchronisation,*
- $\mathsf{idle}^{k}.P$ *as a shorthand for* $\mathsf{idle}.\mathsf{idle}.\ldots.\mathsf{idle}.P$*, where the prefix* idle *appears $k \geq 0$ consecutive times,*
- $\mathsf{if}\ (b)\ \{P\}$ *instead of* $\mathsf{if}\ (b)\ \{P\}\ \mathsf{else}\ \{\mathsf{nil}\}$.

Let $M = E \bowtie_{\mathcal{S}} P$, we write $M \parallel Q$ for $E \bowtie_{\mathcal{S}} (P \parallel Q)$, and $M \backslash c$ for $E \bowtie_{\mathcal{S}} (P \backslash c)$.

We can now finalise our running example.

Example 2 (Cyber component of the CPS *Sys*)**.** *Let us define the cyber component of the CPS Sys described in Example 1. 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 abnormal behaviours of the system. 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 says to Ctrl to keep cooling for other* 5 *time units; otherwise, if the temperature is not above* 10*, the IDS component requires Ctrl to stop the cooling activity.*

$$
\begin{array}{rcl}
Ctrl & = & \mathsf{read}\, s_{\mathrm{t}}(x).\mathsf{if}\ (x > 10)\ \{Cooling\}\ \mathsf{else}\ \{\mathsf{idle}.Ctrl\} \\
Cooling & = & \mathsf{write}\, \mathsf{cool}\langle \mathsf{on} \rangle.\mathsf{idle}^{5}.Check \\
Check & = & \mathsf{snd}\, sync.\mathsf{rcv}\, ins(y).\mathsf{if}\ (y = \mathsf{keep\_cooling}) \\
 & & \{\mathsf{idle}^{5}.Check\}\ \mathsf{else}\ \{\mathsf{write}\, cool\langle \mathsf{off} \rangle.\mathsf{idle}.Ctrl\} \\
IDS & = & \mathsf{rcv}\, sync.\mathsf{read}\, s_{\mathrm{t}}(x).\mathsf{if}\ (x > 10) \\
 & & \{\mathsf{snd}\, alarm\langle \mathsf{high\_temp} \rangle.\mathsf{snd}\, ins\langle \mathsf{keep\_cooling} \rangle. \\
 & & \mathsf{idle}.IDS\}\ \mathsf{else}\ \{\mathsf{snd}\, ins\langle \mathsf{stop} \rangle.\mathsf{idle}.IDS\}\ .
\end{array}
$$

Thus, the whole CPS is defined as:

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

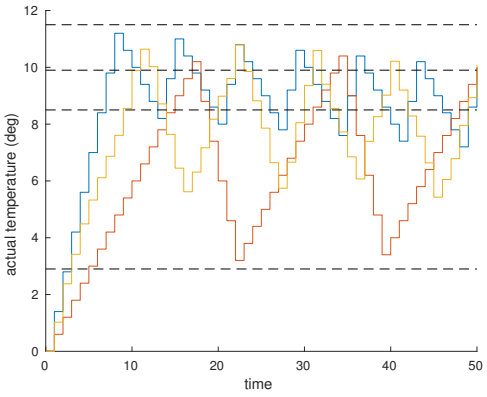

Fig. 2. Three possible evolutions of the CPS of Example 2.

where Env is the physical environment defined in Example 1. We remark that, 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.

A. 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. Definition 5 introduces auxiliary operators on environments.

Definition 5. *Let $E = \langle \xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}}, evol, \xi_{\mathrm{e}}, meas, inv, safe \rangle$.*

- $read\_sensor(E, s) \stackrel{\text{def}}{=} \{\xi(s) : \xi \in meas(\xi_{\mathrm{x}}, \xi_{\mathrm{e}})\}$,
- $update\_act(E, a, v) \stackrel{\text{def}}{=} E[\xi_{\mathrm{u}} \leftarrow \xi_{\mathrm{u}}[a \mapsto v]]$,
- $next(E) \stackrel{\text{def}}{=} \bigcup_{\xi \in evol(\xi_{\mathrm{x}}, \xi_{\mathrm{u}}, \xi_{\mathrm{w}})} \{E[\xi_{\mathrm{x}} \leftarrow \xi]\}$,
- $inv(E) \stackrel{\text{def}}{=} inv(\xi_{\mathrm{x}})$,
- $safe(E) \stackrel{\text{def}}{=} safe(\xi_{\mathrm{x}})$.

The operator $read\_sensor(E, s)$ returns the set of possible measurements detected by sensor s in the environment E; due to the error $\xi_{\mathrm{e}}(s)$ of sensor s, it returns a set of possible values rather than a single value. $update\_act(E, a, v)$ returns the new environment in which the actuator function is updated in such a manner to associate the actuator a with the value v. $next(E)$ returns the set of the next admissible environments reachable from E, by an application of $evol$. $inv(E)$ checks whether the invariant is satisfied by the current values of the state variables (here, abusing notation, we overload the meaning of the function inv). $safe(E)$ checks whether the safety conditions are satisfied by the current values of the state variables.

In Table I, we provide transition rules for processes. Here, the meta-variable λ ranges over labels in the set $\{\mathsf{idle}, \tau, \overline{c}v, cv, a!v, s?v, \lightning p!v, \lightning p?v, \tau{:}p\}$. Rules (Outp), (Inpp) and (Com) serve to model channel communication, on some channel c. Rules (Write) and (Read) denote the writing/reading of some data on the physical device p. Rule ($\lightning$SensWrite$\lightning$) models an *integrity*

TABLE I
LTS FOR PROCESSES

$$\text{(Outp)}\ \frac{-}{\mathsf{timeout}\lfloor \mathsf{snd}\, c\langle v\rangle.P\rfloor Q \xrightarrow{\overline{c}v} P} \qquad \text{(Inpp)}\ \frac{-}{\mathsf{timeout}\lfloor \mathsf{rcv}\, c(x).P\rfloor Q \xrightarrow{cv} P\{^v/_x\}}$$

$$\text{(Com)}\ \frac{P \xrightarrow{\overline{c}v} P' \quad Q \xrightarrow{cv} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \qquad \text{(Par)}\ \frac{P \xrightarrow{\lambda} P' \quad \lambda \neq \mathsf{idle}}{P \parallel Q \xrightarrow{\lambda} P' \parallel Q}$$

$$\text{(Write)}\ \frac{\mu \in \{p, \lightning p\}}{\mathsf{timeout}\lfloor \mathsf{write}\, \mu\langle v\rangle.P\rfloor Q \xrightarrow{\mu!v} P} \qquad \text{(Read)}\ \frac{\mu \in \{p, \lightning p\}}{\mathsf{timeout}\lfloor \mathsf{read}\, \mu(x).P\rfloor Q \xrightarrow{\mu?v} P\{^v/_x\}}$$

$$\text{(}\lightning\text{SensWrite}\lightning\text{)}\ \frac{P \xrightarrow{\lightning s!v} P' \quad Q \xrightarrow{s?v} Q'}{P \parallel Q \xrightarrow{\tau:s} P' \parallel Q'} \qquad \text{(}\lightning\text{ActRead}\lightning\text{)}\ \frac{P \xrightarrow{a!v} P' \quad Q \xrightarrow{\lightning a?v} Q'}{P \parallel Q \xrightarrow{\tau:a} P' \parallel Q'}$$

$$\text{(Res)}\ \frac{P \xrightarrow{\lambda} P' \quad \lambda \notin \{cv, \overline{c}v\}}{P\backslash c \xrightarrow{\lambda} P'\backslash c} \qquad \text{(Rec)}\ \frac{P\{^{\tilde{w}}/_{\tilde{x}}\} \xrightarrow{\lambda} Q \quad H(\tilde{x}) = P}{H\langle \tilde{w}\rangle \xrightarrow{\lambda} Q}$$

$$\text{(Then)}\ \frac{[\![b]\!] = \mathsf{true} \quad P \xrightarrow{\lambda} P'}{\mathsf{if}\ (b)\ \{P\}\ \mathsf{else}\ \{Q\} \xrightarrow{\lambda} P'} \qquad \text{(Else)}\ \frac{[\![b]\!] = \mathsf{false} \quad Q \xrightarrow{\lambda} Q'}{\mathsf{if}\ (b)\ \{P\}\ \mathsf{else}\ \{Q\} \xrightarrow{\lambda} Q'}$$

$$\text{(TimeNil)}\ \frac{-}{\mathsf{nil} \xrightarrow{\mathsf{idle}} \mathsf{nil}} \qquad \text{(Delay)}\ \frac{-}{\mathsf{idle}.P \xrightarrow{\mathsf{idle}} P}$$

$$\text{(Timeout)}\ \frac{-}{\mathsf{timeout}\lfloor \pi.P\rfloor Q \xrightarrow{\mathsf{idle}} Q} \qquad \text{(TimePar)}\ \frac{P \xrightarrow{\mathsf{idle}} P' \quad Q \xrightarrow{\mathsf{idle}} Q'}{P \parallel Q \xrightarrow{\mathsf{idle}} P' \parallel Q'}$$

attack on sensor s, where the controller of s is supplied with a fake value v provided by the attack. Rule (⚡ActRead⚡) models a *DoS attack to the actuator* a, where the update request of the controller is intercepted by the attacker and it never reaches the actuator. Rule (Par) propagates untimed actions over parallel components. Rules (Res), (Rec), (Then) and (Else) are standard. The following four rules model the passage of time. The symmetric counterparts of rules (Com) and (Par) are omitted.

In Table II, we lift the transition rules from processes to systems. Except for rule (Deadlock), all rules have a common premise $inv(E)$: a system can evolve only if the invariant is satisfied. Here, actions, ranged over by α, are in the set $\{\tau, \overline{c}v, cv, \mathsf{idle}, \mathsf{deadlock}, \mathsf{unsafe}\}$. These actions denote: internal activities (τ); logical activities, more precisely, channel transmission ($\overline{c}v$ and cv); the passage of time (idle); 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 (SensReadSec) models the reading of the current data detected at a *secured sensor* s, whereas rule (SensReadUnsec) models the reading of an *unsecured sensor* s. In this case, since the sensor is not secured, the presence of a malicious action $\lightning s!w$ prevents the reading of the sensor. We already said that rule (⚡SensWrite⚡) of Table I models integrity attacks on an unsecured sensor s, however, together with rule (SensReadUnsec), it also serves to model *DoS attacks on an unsecured sensor* s, as the controller of s cannot read its correct value if the attacker is currently supplying a fake value for it.

Rule (⚡SensRead⚡) allows the attacker to read the confidential value detected at an unsecured sensor s. Rule (ActWriteSec) models the writing of a value v on a *secured actuator* a, whereas rule (ActWriteUnsec) models the writing on an *unsecured actuator* a. Again, if the actuator is unsecured, the presence of an attack (capable of performing an action $\lightning a?v$) prevents the correct access to the actuator by the controller. Rule (⚡ActWrite⚡) models an *integrity attack to an unsecured actuator* a, where the attack updates the actuator with a fake value.

Note that our operational semantics ensures a preemptive power to prefixes of the form $\mathsf{write}\lightning p\langle v\rangle$ and $\mathsf{read}\lightning p(x)$ *on unsecured devices* p. This because an attack process can always prevent the regular access to a unsecured physical device (sensor or actuator) by its controller.

Proposition 1 (Attack preemptiveness)**.** *Let* $M = E \bowtie_{\mathcal{S}} P$.

- *If there is* Q *such that* $P \xrightarrow{\lightning s!v} Q$*, with* $s \notin \mathcal{S}$*, then there is no* M' *such that* $M \xrightarrow{\tau} M'$ *by an application of the rule (SensReadUnsec).*
- *If there is* Q *such that* $P \xrightarrow{\lightning a?v} Q$*, with* $a \notin \mathcal{S}$*, then there is no* M' *such that* $M \xrightarrow{\tau} M'$ *by an application of the rule (ActWriteUnsec).*

Rule (Tau) lifts non-observable actions from processes to systems. This includes communications channels and attacks' accesses to unsecured physical devices. A similar lifting occurs in rule (Time) for timed actions, where $\mathrm{next}(E)$ returns the set of possible environments for the next time slot. 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

TABLE II
LTS FOR CPSS

$$\text{(Out)}\ \frac{P \xrightarrow{\overline{c}v} P' \quad \text{inv}(E)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\overline{c}v} E \bowtie_{\mathcal{S}} P'} \qquad \text{(Inp)}\ \frac{P \xrightarrow{cv} P' \quad \text{inv}(E)}{E \bowtie_{\mathcal{S}} P \xrightarrow{cv} E \bowtie_{\mathcal{S}} P'}$$

$$\text{(SensReadSec)}\ \frac{P \xrightarrow{s?v} P' \quad s \in \mathcal{S} \quad \text{inv}(E) \quad v \in \text{read\_sensor}(E,s)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} E \bowtie_{\mathcal{S}} P'}$$

$$\text{(SensReadUnsec)}\ \frac{P \xrightarrow{s?v} P' \quad s \notin \mathcal{S} \quad P \xrightarrow{\lightning s!v} \quad \text{inv}(E) \quad v \in \text{read\_sensor}(E,s)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} E \bowtie_{\mathcal{S}} P'}$$

$$\text{(}\lightning\text{SensRead}\lightning\text{)}\ \frac{P \xrightarrow{\lightning s?v} P' \quad s \notin \mathcal{S} \quad \text{inv}(E) \quad v \in \text{read\_sensor}(E,s)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} E \bowtie_{\mathcal{S}} P'}$$

$$\text{(ActWriteSec)}\ \frac{P \xrightarrow{a!v} P' \quad a \in \mathcal{S} \quad \text{inv}(E) \quad E' = \text{update\_act}(E,a,v)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} E' \bowtie_{\mathcal{S}} P'}$$

$$\text{(ActWriteUnsec)}\ \frac{P \xrightarrow{a!v} P' \quad a \notin \mathcal{S} \quad P \xrightarrow{\lightning a?v} \quad \text{inv}(E) \quad E' = \text{update\_act}(E,a,v)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} E' \bowtie_{\mathcal{S}} P'}$$

$$\text{(}\lightning\text{ActWrite}\lightning\text{)}\ \frac{P \xrightarrow{\lightning a!v} P' \quad a \notin \mathcal{S} \quad \text{inv}(E) \quad E' = \text{update\_act}(E,a,v)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} E' \bowtie_{\mathcal{S}} P'}$$

$$\text{(Tau)}\ \frac{(P \xrightarrow{\tau} P') \vee (P \xrightarrow{\tau:p} P' \ p \notin \mathcal{S}) \quad \text{inv}(E)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} E \bowtie_{\mathcal{S}} P'} \qquad \text{(Deadlock)}\ \frac{\neg\, \text{inv}(E)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\mathsf{deadlock}} E \bowtie_{\mathcal{S}} P}$$

$$\text{(Time)}\ \frac{P \xrightarrow{\mathsf{idle}} P' \quad E \bowtie_{\mathcal{S}} P \xrightarrow{\tau} \quad \text{inv}(E) \quad E' \in \text{next}(E)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\mathsf{idle}} E' \bowtie_{\mathcal{S}} P'} \qquad \text{(Safety)}\ \frac{\neg\, \text{safe}(E) \quad \text{inv}(E)}{E \bowtie_{\mathcal{S}} P \xrightarrow{\mathsf{unsafe}} E \bowtie_{\mathcal{S}} P}$$

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.

Now, having defined the actions that can be performed by a system, 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 $\xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n}$. In the following, we will use the function $\#\mathsf{idle}(t)$ to get the number of occurrences of the action idle in t.

The notion of trace allows us to provide a formal definition of soundness for CPSs: 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, both actions* deadlock *and* unsafe *never occur in t.*

In our security analysis, we will focus on sound CPSs. For instance, Proposition 2 says that our running example *Sys* is sound and it never transmits on the channel $alarm$.

Proposition 2. *Let Sys be the CPS defined in Example 2. If Sys $\xrightarrow{t}$ Sys′, for some trace t=$\alpha_1 \dots \alpha_n$, then $\alpha_i \in \{\tau, \mathsf{idle}\}$, 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 laying in the real interval $[1-\delta, 1+\delta]$, where δ is the uncertainty.

Proposition 3. *Let Sys be the CPS defined in Example 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 graphic in Figure 3 collects a campaign of 100 simulations, lasting 250 time slots each, showing that the value of the state variable *temp* when the cooling system is turned on (resp., off) lays in the interval $(9.9, 11.5]$ (resp., $(2.9, 8.5]$); these bounds are represented by the dashed horizontal lines.

B. Behavioural semantics

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 $\overset{\alpha}{\Longrightarrow}$ means $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$, and finally $\overset{\hat{\alpha}}{\Longrightarrow}$ denotes $\Rightarrow$ if $\alpha = \tau$ and $\overset{\alpha}{\Longrightarrow}$ otherwise. Given a trace $t = \alpha_1 \dots \alpha_n$, we write $\overset{t}{\longrightarrow}$ for $\xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_n}$, and $\overset{\hat{t}}{\Longrightarrow}$ as an abbreviation for $\overset{\widehat{\alpha_1}}{\Longrightarrow} \dots \overset{\widehat{\alpha_n}}{\Longrightarrow}$.

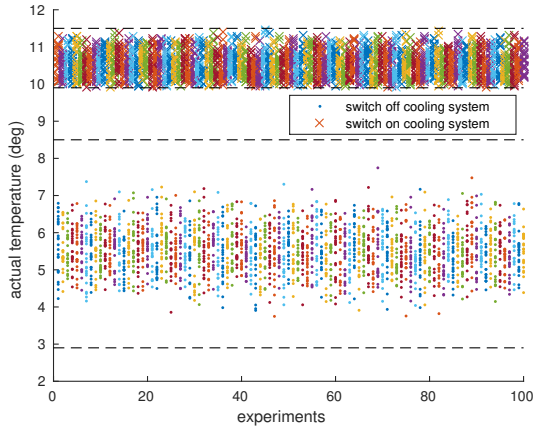

Fig. 3. Simulations of the CPS of Example 2

Definition 7 (Trace preorder)**.** *We write $M \sqsubseteq N$ if whenever $M \xrightarrow{t} M'$, for some t, there is N' such that $N \stackrel{\hat{t}}{\Longrightarrow} N'$.*

Remark 1. *Unlike standard trace semantics, our trace preorder is able to observe deadlock thanks to the presence of the rule (Deadlock) and the special action* deadlock*: if $M \sqsubseteq N$ and M eventually deadlocks then also N must eventually deadlock.*

As we are interested in examining timing aspects of attacks, such as beginning and duration, we propose a timed variant of $\sqsubseteq$ up to (a possibly infinite) time interval. Intuitively, we write $M \sqsubseteq_{m..n} N$ if the CPS N simulates the execution traces of M, except for the time interval $m..n$.

Definition 8 (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 $\#\mathsf{idle}(t) = m - 1$, such that $M \xrightarrow{t}$ and $N \stackrel{\hat{t}}{\not\Longrightarrow}$;*
- *n is the infimum element of $\mathbb{N}^+ \cup \{\infty\}$, $n \geq m$, such that whenever $M \xrightarrow{t_1} M'$, with $\#\mathsf{idle}(t_1) = n - 1$, there is t_2, with $\#\mathsf{idle}(t_1) = \#\mathsf{idle}(t_2)$, such that $N \xrightarrow{t_2} N'$, for some N', and $M' \sqsubseteq N'$.*

In Definition 8, 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) after the time slot n 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.

Finally, note that we could have equipped CCPSA with a *(bi)simulation-based* behavioural semantics rather than a trace-based one, as done in our companion paper [15] for a core of CCPSA with no security features; however, our trace semantics is simpler than (bi)simulation and it is sensitive to deadlocks of CPSs. Thus, it is fully adequate for the purposes of this paper.

III. Cyber-Physical Attacks

In this section, we use CCPSA to formalise a *threat model* where attacks can manipulate sensor and/or actuator signals in order to drive a *sound* CPS into an undesired state [26]. An attack may have different levels of access to physical devices depending on the model assumed. 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 attacks can affect a CPS [4]. For simplicity, in this paper we don't represent attacks on communication channels as our focus is on attacks to physical devices.

The syntax of our cyber-physical attack is a slight restriction of that for processes: in terms of the form $\mathsf{timeout}\lfloor \pi.P \rfloor Q$, we require $\pi \in \{\mathsf{write}\, \lightning p\langle v \rangle, \mathsf{read}\, \lightning p(x)\}$. Thus, we provide a syntactic way to distinguish attacks from genuine processes.

Definition 9 (Honest system)**.** *A CPS $E \bowtie_{\mathcal{S}} P$ is* honest *if P is honest, where P is honest if it does not contain prefixes of the form* $\mathsf{write}\, \lightning p\langle v \rangle$ *or* $\mathsf{read}\, \lightning p(x)$.

We group cyber-physical 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 $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$* if:

- all possible malicious activities of A coincide with those contained in $\mathcal{I}$;
- the first of those actions may occur in the m-th time slot (A admits a run acting at time m) but not before;
- the last of those actions may occur in the n-th time slot (A admits a run acting at time n) but not after;
- 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 $m..n$;
- C is a total function: if $C(\iota) = \emptyset$, for some $\iota \in \mathcal{I}$, then no attacks of class C can achieve the malicious activity ι.

Definition 10 (Class of attacks)**.** *Let $\mathcal{I} = \{\lightning p? : p \in \mathcal{S} \cup \mathcal{A}\} \cup \{\lightning p! : p \in \mathcal{S} \cup \mathcal{A}\}$ 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 *$C \in [\mathcal{I} \to \mathcal{P}(m..n)]$ is a total function such that for any attack A of class C we have:*

(i) $C(\iota) = \{k : A \xrightarrow{t} \xrightarrow{\iota v} A' \wedge k = \#\mathsf{idle}(t) + 1\}$, *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 [7], [18], we can say that an attack A affects a *sound* CPS M if the execution of the composed system $M \parallel A$ differs from that of the original system M, in an observable manner. Basically, a cyber-physical 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 cannot be reproduced by M; here the attack affects the *integrity* of the system behaviour.

Definition 11 (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 $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$, 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 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* [10]. 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 $M \parallel A \xrightarrow{t} \xrightarrow{\mathsf{deadlock}}$, for some trace t, the attack A is called *lethal*, as it is capable to halt (deadlock) the CPS M. This is obviously a permanent attack.

Note that, according to Definition 11, 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.

In the following, 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.

In the rest of this section, we present a number of different attacks to the CPS *Sys* described in Example 2.

Example 3. *Consider the following* DoS/Integrity attack *on the (controller of) the actuator cool, of class $C \in [\mathcal{I} \to \mathcal{P}(m..m)]$ with $C(\lightning cool?) = C(\lightning cool!) = \{m\}$ and $C(\iota) = \emptyset$, for $\iota \notin \{\lightning cool?, \lightning cool!\}$; we call the attack A_m:*

$$\mathsf{idle}^{m-1}.\mathsf{timeout}\lfloor \mathsf{read}\lightning cool(x).\mathsf{if}(x{=}\mathsf{off})\{\mathsf{write}\lightning cool\langle\mathsf{off}\rangle\}\rfloor.$$

Here, the attack A_m operates exclusively in the m-th time slot, when it tries to steal the cooling command (on or off) coming from the controller, and fabricates a fake command to turn off the cooling system. In practice, if the controller sends a command to turn off the coolant, nothing bad will happen as the attack will put the same message back. When the controller sends (in the m-th time slot) a command to turn the cooling on, 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 $temp > 9.9$); 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 4 time units the system will 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.*

Proposition 4. *Let Sys be the CPS defined in Example 2, and A_m be the attack defined in Example 3. Then,*

- *Sys $\parallel A_m \sqsubseteq$ Sys, for $m \leq 8$,*
- *Sys $\parallel A_m \not\sqsubseteq_{m+4..\infty}$ Sys, for $m > 8$.*

In this case, the *IDS* component of *Sys* is effective enough to detect the attack with only one time unit delay.

Example 4. *Consider the following* DoS/Integrity attack *to the (controller of) sensor s_{t}, of class $C \in [\mathcal{I} \to \mathcal{P}(2..\infty)]$ such that $C(\lightning s_{\mathrm{t}}?) = \{2\}$, $C(\lightning s_{\mathrm{t}}!) = 2..\infty$ and $C(\iota) = \emptyset$, for $\iota \notin \{\lightning s_{\mathrm{t}}!, \lightning s_{\mathrm{t}}?\}$:*

$$\begin{array}{rcl} A & = & \mathsf{idle}.\mathsf{timeout}\lfloor \mathsf{read}\lightning s_{\mathrm{t}}(x).B\langle x\rangle\rfloor \\ B(y) & = & \mathsf{timeout}\lfloor \mathsf{write}\lightning s_{\mathrm{t}}\langle y\rangle.\mathsf{idle}.B\langle y\rangle\rfloor B\langle y\rangle \ . \end{array}$$

Here, the attack A does the following actions in sequence: (i) she sleeps for one time unit, (ii) in the following time slot, she reads the current temperature v at sensor s_{t}, and (iii) for the rest of her life, she keeps sending the same temperature v to the controller of s_{t}. In the presence of this attack, the process Ctrl never activates the Cooling component (and, hence, nor the IDS component, which is the only one which could send an alarm) as it will always detect a temperature below 10. Thus, the compound system Sys $\parallel A$ will move to an unsafe state until the invariant will be violated and the system will deadlock. Indeed, in the worst scenario, after $\lceil \frac{9.9}{1+\delta} \rceil = \lceil \frac{9.9}{1.4} \rceil = 8$ idle*-actions (in the 9-th time slot) the value of $temp$ will be above 9.9, and after further 5* idle*-actions (in the 14-th time slot) the system will violate the safety conditions emitting an* unsafe *action. After $= \lceil \frac{50}{1.4} \rceil = 36$* idle*-actions, in the 37-th time slot, the invariant may be broken because the state variable $temp$ may reach 50.4 degrees, and the system will also emit a* deadlock *action. Thus, Sys $\parallel A \not\sqsubseteq_{14..\infty}$ Sys. This is a* lethal *attack, as it causes a shut down of the system. It is also a* stealthy attack *as it remains unnoticed until the end.*

In this attack, the IDS component is completely ineffective as the sensor used by the component is compromised, and there is no way for the IDS to understand whether the sensor is under attack. A more sophisticated IDS might have a representation of the plant to recognise abnormal evolutions of the sensed temperature. In such case, the IDS might switch on a second sensor, hoping that this one has not been compromised yet. Another possibility for the designer of the CPS is to secure the sensor. Although this is not always possible,

as encryption/decryption of all packets depends on energy constraints of the device.

Our semantics ensures that secured devices cannot be attacked, as stated by the following proposition.

Proposition 5. *Let $M = E \bowtie_{\mathcal{S}} P$ be an honest and sound CPS. Let $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$, with $\{p : C(\lightning p?) \cup C(\lightning p!) \neq \emptyset\} \subseteq \mathcal{S}$. Then $M \parallel A \sqsubseteq M$, for any attack A of class C.*

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

Example 5. *Consider the following DoS/Integrity attack to the controller of sensor $s_{\rm t}$, of class $C \in [\mathcal{I} \to \mathcal{P}(1..n)]$, for $n > 0$, with $C(\lightning s_{\rm t}!) = C(\lightning s_{\rm t}?) = 1..n$ and $C(\iota) = \emptyset$, for $\iota \notin \{\lightning s_{\rm t}!, \lightning s_{\rm t}?\}$:*

$$A_n \;=\; \mathsf{timeout}\lfloor \mathsf{read}\,\lightning s_{\rm t}(x).\mathsf{timeout}\lfloor \{\mathsf{write}\,\lightning s_{\rm t}\langle x{-}2\rangle.\ \mathsf{idle}.A_{n-1}\rfloor A_{n-1}\rfloor A_{n-1}$$

with $A_0 = \mathsf{nil}$. In this attack, for n consecutive time slots, A_n sends to the controller the current sensed temperature decreased by an offset 2. The effect of this attack on the system depends on the duration n of the attack itself:

- *for $n \leq 8$, the attack is harmless as the variable $temp$ may not reach a (critical) temperature above 9.9;*
- *for $n = 9$, the variable $temp$ might reach a temperature above 9.9 in the 9-th 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 slots 14 and 15, but no alarm will be fired.*
- *for $n \geq 10$, the system may get into an unsafe state in the time slot 14 and in the following $n - 7$ time slots; this is not a* stealthy attack *as the IDS will fire the alarm at most two time slots later (in the 16-th time slot); this is a* temporary attack *which ends in the time slot $n + 7$.*

Proposition 6. *Let Sys be the CPS defined in Example 2, and A_n be the attack defined in Example 5. Then:*

- *Sys $\parallel A_n \sqsubseteq$ Sys, for $n \leq 8$,*
- *Sys $\parallel A_n \not\sqsubseteq_{14..15}$ Sys, for $n = 9$,*
- *Sys $\parallel A_n \not\sqsubseteq_{14..n+7}$ Sys, for $n \geq 10$.*

A. 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 a class C' that is somehow "weaker" than a given class C.

Definition 12. *Let $C_1 \in [\mathcal{I} \to \mathcal{P}(m_1..n_1)]$ and $C_2 \in [\mathcal{I} \to \mathcal{P}(m_2..n_2)]$ 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 $C_1 \preceq C_2$, if $C_1(\iota) \subseteq C_2(\iota)$ for any $\iota \in \mathcal{I}$.*

Intuitively, if $C_1 \preceq C_2$ then: (i) the attacks of class C_1 might achieve fewer malicious activities than any attack of class C_2 (formally, there may be $\iota \in \mathcal{I}$ such that $C_1(\iota) = \emptyset$ and $C_2(\iota) \neq \emptyset$); (ii) for those malicious activities $\iota \in \mathcal{I}$ 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 $C' \preceq C$. We will provide a similar condition for attack vulnerability: let M be a CPS vulnerable to $Top(C)$ in the time interval $m_1..n_1$; then, for any attack A of class C', with $C' \preceq C$, 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 cyber-physical 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 $rnd()$ 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 $\mathsf{if}\ (rnd(\{\mathsf{true}, \mathsf{false}\}))\ \{P\}\ \mathsf{else}\ \{Q\}$. Thus, let $\iota \in \{\lightning p? : p \in \mathcal{S} \cup \mathcal{A}\} \cup \{\lightning p! : p \in \mathcal{S} \cup \mathcal{A}\}$, $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+ \cup \{\infty\}$, with $m \leq n$, and $\mathcal{T} \subseteq m..n$, we define the attack process $Att(\iota, k, \mathcal{T})$ 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 $\mathcal{T}$. Formally,

$$\begin{array}{l} Att(\lightning p?, k, \mathcal{T}) = \\ \quad \mathsf{if}\ (k \in \mathcal{T})\ \{(\mathsf{timeout}\lfloor \mathsf{read}\,\lightning p?(x).Att(\lightning p?, k, \mathcal{T})\rfloor \\ \quad\quad Att(\lightning p?, k{+}1, \mathcal{T})) \oplus \mathsf{idle}.Att(\lightning p?, k{+}1, \mathcal{T})\}\ \mathsf{else} \\ \quad\quad \mathsf{if}\ (k < \sup(\mathcal{T}))\ \{\mathsf{idle}.Att(\lightning p?, k{+}1, \mathcal{T})\}\ \mathsf{else}\ \{\mathsf{nil}\} \\ Att(\lightning p!, k, \mathcal{T}) = \\ \quad \mathsf{if}\ (k \in \mathcal{T})\ \{(\mathsf{timeout}\lfloor \mathsf{write}\,\lightning p!\langle rnd(\mathbb{R})\rangle.Att(\lightning p!, k, \mathcal{T})\rfloor \\ \quad\quad Att(\lightning p!, k{+}1, \mathcal{T})) \oplus \mathsf{idle}.Att(\lightning p!, k{+}1, \mathcal{T})\}\ \mathsf{else} \\ \quad\quad \mathsf{if}\ (k < \sup(\mathcal{T}))\ \{\mathsf{idle}.Att(\lightning p!, k{+}1, \mathcal{T})\}\ \mathsf{else}\ \{\mathsf{nil}\}\ . \end{array}$$

Note that for $\mathcal{T} = \emptyset$ we assume $\sup(\mathcal{T}) = -\infty$.

We can now use the definition above to formalise the notion of most powerful attack of a given class C.

Definition 13 (Top attacker)**.** *Let $C \in [\mathcal{I} \to \mathcal{P}(m..n)]$ be a class of attacks. We define*

$$Top(C) \;=\; \textstyle\prod_{\iota \in \mathcal{I}} Att(\iota, 1, C(\iota))$$

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

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

Theorem 1 (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 $C' \preceq C$.*

- *If $M \parallel Top(C) \sqsubseteq M$ then $M \parallel A \sqsubseteq M$.*
- *If $M \parallel Top(C) \not\sqsubseteq_{m_1..n_1} M$ then either $M \parallel A \not\sqsubseteq M$ or $M \parallel A \not\sqsubseteq_{m_2..n_2} M$, with $m_2..n_2 \subseteq m_1..n_1$.*

Proof (sketch). The top attacker $Top(C)$ can mimic any execution trace of any attack A of class C', with $C' \preceq C$ (for

a formal proof of this statement we refer the reader to [16]). Thus, if $M \parallel A \xrightarrow{t}$, for some trace t, then $M \parallel Top(C) \xrightarrow{t}$ as well.

For any M and A, either $M \parallel A \sqsubseteq M$ or $M \parallel A \sqsubseteq_{m_2..n_2} M$, for some m_2 and n_2 ($m_2 = 1$ and $n_2 = \infty$ if the two systems are completely unrelated). Suppose by contradiction that $M \parallel A \not\sqsubseteq M$ and $M \parallel A \sqsubseteq_{m_2..n_2} M$, with $m_2..n_2 \not\subseteq m_1..n_1$. There are two cases: either $n_1 = \infty$ or $n_1 \in \mathbb{N}^+$.

If $n_1 = \infty$ then $m_2 < m_1$. Since $M \parallel A \sqsubseteq_{m_2..n_2} M$, by Definition 8 there is a trace t, with $\#\mathsf{idle}(t) = m_2 - 1$, such that $M \parallel A \xrightarrow{t}$ and $M \stackrel{\hat{t}}{\not\Longrightarrow}$. Since $Top(C)$ can mimic any execution trace of any attack A of class C', with $C' \preceq C$, this entails $M \parallel Top(C) \xrightarrow{t}$. Since $M \stackrel{\hat{t}}{\not\Longrightarrow}$ and $\#\mathsf{idle}(t) = m_2 - 1 < m_1$, this contradicts $M \parallel Top(C) \sqsubseteq_{m_1..n_1} M$.

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. □

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 $C' \preceq C$, is not a lethal attack for M. If $Top(C)$ is not a permanent attack for M then any attack A of class C', with $C' \preceq C$, is not a permanent attack for M.*

Remark 2. *The reader may wonder about the scalability of the verification method proposed in Theorem 1. We would like to recall that in our companion paper [15] we developed a* (bi)simulation (pre)congruence *for a simpler version of the calculus where security features have been completely stripped off. For simplicity, in the current paper, we adopted as main behavioural semantics trace preorder instead of simulation preorder. We believe that switching to a simulation precongruence, preserved by parallel composition, would allow us to scale our verification method to bigger systems (i.e., we believe that using simulation precongruence instead of trace preorder would allow our verification method to enjoy scalability "for free"). However, this will need to be checked thoroughly in order to consider the more complex version of the calculus with security features that we focus on here.*

In the following, we provide a small example to explain how Theorem 1 could be used to infer attack tolerance/vulnerability with respect to an entire class of attacks.

Example 6. *Consider the CPS Sys of Example 2 and a class of attacks C, such that $C(\lightning s_{\mathrm{t}}!) = C(\lightning s_{\mathrm{t}}?) = \{13\}$, and $C(\iota) = \emptyset$, for $\iota \notin \{\lightning s_{\mathrm{t}}!, \lightning s_{\mathrm{t}}?\}$. Attacks of class C can read/tamper with the sensor s_{t} (exclusively) in the time slot* 13 *(i.e., in the time interval* 13..13*). In Proposition 7 we will show that:*

- *$Top(C)$ may never lead the system Sys in deadlock,*
- *$Top(C)$ may drag the system Sys in an unsafe state only in the time interval* 17..19 *(no alarms are possible).*

As a consequence, each attack of class C (or weaker than C) cannot achieve more than that. Let us provide three different attacks of classes weaker than or equal to C.

Consider the attack:

$$A_1 \quad = \quad \mathsf{idle}^{12}.\mathsf{timeout}\lfloor \mathsf{write}\lightning s_{\mathrm{t}}\langle 11\rangle \rfloor \ .$$

This attack forces the activation of the cooling system without any consequence to the observable behaviour of the whole system (see Proposition 7). The attack belongs to a class C', with $C' \preceq C$, such that $C'(\lightning s_{\mathrm{t}}!) = \{13\}$ and $C'(\iota) = \emptyset$, for $\iota \neq \lightning s_{\mathrm{t}}!$.

Consider a different attack:

$$A_2 \quad = \quad \mathsf{idle}^{12}.\mathsf{timeout}\lfloor \mathsf{write}\lightning s_{\mathrm{t}}\langle 9\rangle \rfloor \ .$$

This attack is of the same class C'. However, it may delay the activation of the cooling system, and due to such delay the CPS under attack may get into an unsafe state in the time interval 17..19 *(see Proposition 7).*

Finally, consider the attack:

$$A_3 \quad = \quad \mathsf{idle}^{12}.\mathsf{timeout}\lfloor \mathsf{read}\lightning s_{\mathrm{t}}(x).\mathsf{if}\ (x{=}10.5)\ \{\mathsf{timeout}\lfloor \mathsf{write}\lightning s_{\mathrm{t}}\langle 9\rangle \rfloor\}\rfloor \ .$$

This is an attack of class C which delays the activation of the cooling system as well; however, this delay may only occur when the temperature is quite low. As a consequence, the system under attack may get into an unsafe state only in the time slot 18*, i.e., the time interval* 18..18*, (again, see Proposition 7).*

The following proposition formalises the statements contained in Example 6.

Proposition 7. *Let Sys be the CPS of Example 2 and C be a class of attacks such that $C(\lightning s_{\mathrm{t}}!) = C(\lightning s_{\mathrm{t}}?) = \{13\}$, and $C(\iota) = \emptyset$, for $\iota \notin \{\lightning s_{\mathrm{t}}!, \lightning s_{\mathrm{t}}?\}$. Then,*

- *$Sys \parallel Top(C) \sqsubseteq_{17..19} Sys$,*
- *$Sys \parallel A_1 \sqsubseteq Sys$,*
- *$Sys \parallel A_2 \sqsubseteq_{17..19} Sys$,*
- *$Sys \parallel A_3 \sqsubseteq_{18..18} Sys$.*

IV. Impact of an attack

In the previous section, we have grouped cyber-physical attacks by focussing on the physical devices under attack and the timing aspects of the attack (Definition 10). Then, we have provided a formalisation of when a CPS should be considered tolerant/vulnerable to an attack (Definition 11). In this section, we show that it is important not only to demonstrate the tolerance (or vulnerability) of a CPS with respect to certain attacks, but also to evaluate the disruptive impact of those attacks on the target CPS [9].

The goal of this section is thus twofold: to provide a *metric* to estimate the impact of a successful attack on a CPS, and to investigate possible quantifications of the chances for an attack of being successful when attacking a CPS.

As to the metric, we focus on the ability that an attack may have to drag a CPS out of the correct behaviour modelled by its evolution map, with the given uncertainty. Recall that $evol$ is *monotone* with respect to the uncertainty. Thus, an increase of the uncertainty may translate into a widening of the range of the possible behaviours of the CPS.

In the following, for $M = E \bowtie_{\mathcal{S}} P$, we write $M[\psi \leftarrow \psi']$ to mean $E[\psi \leftarrow \psi'] \bowtie_{\mathcal{S}} P$.

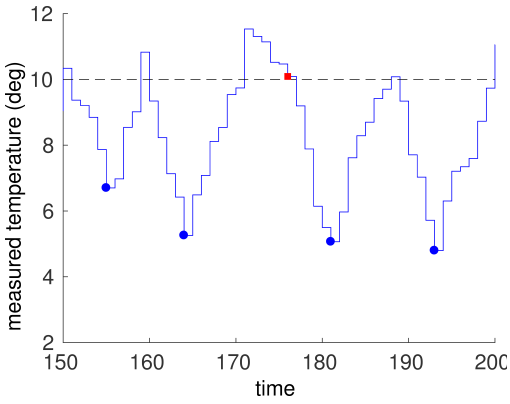

Fig. 4. Simulation of $Sys[\delta \leftarrow \frac{29}{30}]$.

Proposition 8 (Monotonicity)**.** *Let $M = E \bowtie_{\mathcal{S}} P$ be an honest and sound CPS, and ξ_{w} the uncertainty of E. If $\xi_{\mathrm{w}} \leq \xi'_{\mathrm{w}}$ and $M \xrightarrow{t} M'$ then $M[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}] \xrightarrow{t} M'[\xi_{\mathrm{w}} \leftarrow \xi'_{\mathrm{w}}]$.*

However, a wider uncertainty in the model doesn't always correspond to a widening of the possible behaviours of the CPS. In fact, this depends on the *intrinsic tolerance* of a CPS with respect to changes in the uncertainty function.

Definition 14 (System ξ-tolerance)**.** *An honest and sound CPS $M = E \bowtie_{\mathcal{S}} P$, where ξ_{w} is the uncertainty of E, is* ξ-tolerant*, for $\xi \in \mathbb{R}^{\hat{\mathcal{X}}}$ and $\xi \geq 0$, if*

$$\xi \;=\; \sup\left\{\xi' :\; M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\eta] \sqsubseteq M,\ \textit{for any } 0 \leq \eta \leq \xi'\right\}.$$

Intuitively, if a CPS M has been designed with a given uncertainty ξ_{w}, but M is actually ξ-tolerant, with $\xi > 0$, then the uncertainty ξ_{w} is somehow underestimated: the real uncertainty of M is given by $\xi_{\mathrm{w}}{+}\xi$. This information is quite important when trying to estimate the impact of an attack on a CPS. In fact, if a system M has been designed with a given uncertainty ξ_{w}, but M is actually ξ-tolerant, with $\xi > 0$, then an attack has (at least) a "room for manoeuvre" ξ to degrade the whole CPS without being observed (and hence detected).

Let Sys be the CPS of Example 2. In the rest of the section, with an abuse of notation, we will write $Sys[\delta \leftarrow \gamma]$ to denote Sys where the uncertainty of the variable $temp$ is γ.

Example 7. *The CPS Sys of Example 2 is $\frac{1}{20}$-tolerant. This is because $\sup\left\{\xi' : Sys[\delta \leftarrow \delta{+}\eta] \sqsubseteq Sys,\ \textit{for } 0 \leq \eta \leq \xi'\right\}$ is equal to $\frac{1}{20}$. Since $\delta + \xi = \frac{8}{20} + \frac{1}{20} = \frac{9}{20}$, the proof of this statement relies on the following proposition.*

Proposition 9. *Let Sys be the CPS of Example 2. Then:*

- $Sys[\delta \leftarrow \gamma] \;\sqsubseteq\; Sys$*, for* $\gamma \in (\frac{8}{20}, \frac{9}{20})$,
- $Sys[\delta \leftarrow \gamma] \;\not\sqsubseteq\; Sys$*, for* $\gamma > \frac{9}{20}$.

Figure 4 shows an evolution of $Sys[\delta \leftarrow \frac{29}{30}]$: the red box denotes a violation of the safety conditions because the cooling cycle wasn't sufficient to drop the (sensed) temperature below 10 (here, the controller imposes 5 further time units of cooling).

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

Definition 15 (Impact)**.** *Let $M = E \bowtie_{\mathcal{S}} P$ be an honest and sound CPS, where ξ_{w} is the uncertainty of E. We say that an attack A has* definitive impact *ξ on the system M if*

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

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

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

Intuitively, the impact of an attacker A on a system M measures the uncertainty 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', with $C' \preceq C$.

Theorem 2 (Top attacker's impact)**.** *Let M be an honest and sound CPS, and C an arbitrary class of attacks. Let A be an attack of class C', with $C' \preceq C$.*

- *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 $\xi' \leq \xi$.*

Proof (sketch). Consider the case of the definitive impact. The top attacker $Top(C)$ can mimic any execution trace of any attack A of class C', with $C' \preceq C$ (for a formal proof of this statement we refer the reader to [16]). As a consequence, $M \parallel A \xrightarrow{t}$ entails $M \parallel Top(C) \xrightarrow{t}$. This implies $M \parallel A \sqsubseteq M \parallel Top(C)$. Thus, if $M \parallel Top(C) \sqsubseteq M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\xi]$, for $\xi \in \mathbb{R}^{\hat{\mathcal{X}}}$, $\xi > 0$, then, by transitivity of $\sqsubseteq$, it follows that $M \parallel A \sqsubseteq M[\xi_{\mathrm{w}} \leftarrow \xi_{\mathrm{w}}{+}\xi]$.

The proof in the case of the pointwise impact is by contradiction. □

In order to provide more intuition about what the metrics given in Definition 15 measure, we give a couple of examples.

Example 8. *Let us consider the attack A_n of Example 5, for $n = 10$. Then, A_{10} has a definitive impact of $\frac{7}{30}$ to the CPS Sys defined in Example 2. Formally,*

$$\frac{7}{30} = \inf\left\{\xi' \;:\; \xi' > 0 \;\wedge\; Sys \parallel A_{10} \;\sqsubseteq\; Sys[\delta \leftarrow \delta + \xi']\right\}\ .$$

Here, the attack may drag the system into an unsafe state during the time interval $14..17$ (temporary attack). As the attack can affect the system for only 4 time slots, its impact is relatively low: the temperature may rise of at most $1.6\overline{3}$ degrees per time slot, instead of 1.4.

Technically, since $\delta + \xi = \frac{12}{30} + \frac{7}{30} = \frac{19}{30}$, the proof of what stated in this example relies on the following proposition.

Proposition 10. *Let Sys be the CPS defined in Example 2, and A_{10} be the attack defined in Example 8. Then:*

- $Sys \parallel A_{10} \not\sqsubseteq Sys[\delta \leftarrow \gamma]$, *for* $\gamma \in (\frac{12}{30}, \frac{19}{30})$,
- $Sys \parallel A_{10} \sqsubseteq Sys[\delta \leftarrow \gamma]$, *for* $\gamma > \frac{19}{30}$.

The following example shows that the attack provided in Example 4 has much stronger impact on the CPS *Sys*.

Example 9. *Let us consider the attack A of Example 4. As this attack prevents the activation of the cooling system, the temperature will keep growing until the CPS Sys gets into an unsafe state and eventually deadlocks. 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,*

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

Intuitively, in order to deadlock the system without being noticed, the attack must induce a rapid increase of the temperature: 9.9 *degrees per time slot, instead of* 1.4.

Technically, since $\delta+\xi = 0.4+8.5 = 8.9$, *the proof of what is stated in this example relies on the following proposition.*

Proposition 11. *Let Sys be the CPS defined in Example 2, and A be the attack defined in Example 9. Then:*

- $Sys \parallel A \not\sqsubseteq Sys[\delta \leftarrow \gamma]$, *for* $\gamma \in (0.4, 8.9)$,
- $Sys \parallel A \sqsubseteq Sys[\delta \leftarrow \gamma]$, *for* $\gamma > 8.9$.

Definition 15 provided an instrument to estimate the impact of a successful attack. However, there is at least another question that a CPS designer could ask: "Is there a way to estimate the chances that an attack will be successful during the execution of my CPS?" To paraphrase in a more operational manner: how many execution traces of my CPS are prone to be attacked by a specific attack?

For instance, consider again the simple attack A_m proposed in Example 3:

$$\mathsf{idle}^{m-1}.\mathsf{timeout}\lfloor \mathsf{read}\,\lightning cool(x).\mathsf{if}\ (x{=}\mathsf{off})\ \{\mathsf{write}\,\lightning cool\langle \mathsf{off}\rangle\}\rfloor.$$

Here, in the m-th time slot the attack drops a command to turn on the cooling. The attack is very quick and condensed in a single time slot. The question is: what are the chances of success of such a quick attack?

Figure 5 provides a representation of an experiment in MATLAB where we launched 10000 executions of our CPS in isolation, lasting 700 time units each. From the aggregated data contained in this graphic, we note that after a transitory phase (whose length depends on several things: the uncertainty δ, the initial state of the system, the length of the cooling activity, etc.) that lasts around 300 time slots, the rate of success of the attack A_m is around 10%. The reader may wonder why exactly the 10%. This depends on the periodicity of our CPS, as in average the cooling is activated every 10 time slots.

This example shows that, as pointed out in [10], the effectiveness of a cyber-physical attack depends on the information the attack has about the functionality of the whole CPS. For instance, if the attacker were not aware of the exact periodicity of the CPS, she might try, if possible, to repeat the attack on

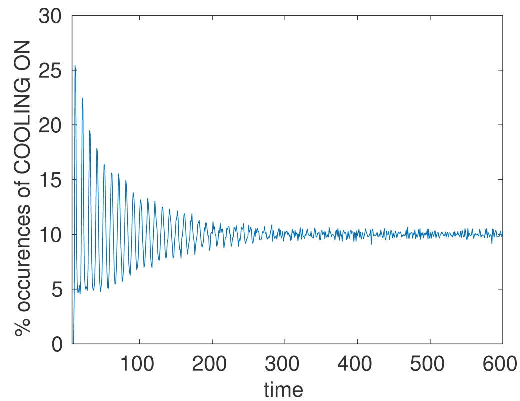

Fig. 5. A quantitative analysis of the attack of Example 3.

more consecutive time slots. In this case, the left graphic of Figure 6 says that the rate of success of the attack increases linearly with the length of the attack itself (data obtained by attacking the CPS after the transitory period). Thus, if the attack of Example 3 were iterated for 10 time slots, say

$$\begin{array}{rcl} A^{10}_m & = & \mathsf{idle}^{m-1}.B_{10} \\ B_i & = & \mathsf{timeout}\lfloor \mathsf{read}\,\lightning cool(x).\mathsf{if}\ (x = \mathsf{off}) \\ & & \{\mathsf{write}\,\lightning cool\langle \mathsf{off}\rangle.\mathsf{idle}.B_{i-1}\}\rfloor B_{i-1},\ \text{for } 1 \leq i \leq 10 \end{array}$$

with $B_0 = \mathsf{nil}$, the rate of success would be almost 100%.

Finally, consider a generalisation of the attack of Example 5:

$$\begin{array}{rcl} A^k_0 & = & \mathsf{nil} \\ A^k_n & = & \mathsf{timeout}\lfloor \mathsf{read}\,\lightning s_{\mathrm{t}}(x).\mathsf{timeout}\lfloor \mathsf{write}\,\lightning s_{\mathrm{t}}\langle x{-}k\rangle. \\ & & \mathsf{idle}.A^k_{n-1}\rfloor A^k_{n-1}\rfloor A^k_{n-1} \end{array}$$

for $1 \leq n \leq 15$ and $2 \leq k \leq 10$. Here, the attack decreases the sensed temperature of an offset k. Now, suppose to launch this attack after, say, 300 time slots (i.e., after the transitory phase). Formally, we define the attack: $B^k_n = \mathsf{idle}^{300}.A^k_n$. In this case, the right graphic of Figure 6 provides a graphical representation of the percentage of alarms on 5000 execution traces lasting 100 time units each. Thus, for instance, an attack lasting $n = 8$ time units with an offset $k = 5$ affects around 40% of the execution traces of the CPS.

These two examples show 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. This would require a probabilistic extension of our approach, which we will investigate in future work.

V. Conclusions, Related and Future Work

We have provided formal theoretical foundations to reason about, and statically detect, attacks to physical devices of CPSs. To that end, we have proposed a hybrid process calculus, called `CCPSA`, as a formal *specification language* to model physical and cyber components of CPSs as well as cyber-physical attacks. Based on `CCPSA` and its labelled transition semantics, we have formalised a threat model for CPSs by grouping attacks in classes, according to the target physical devices and two timing parameters: begin and duration of the attacks. Then,

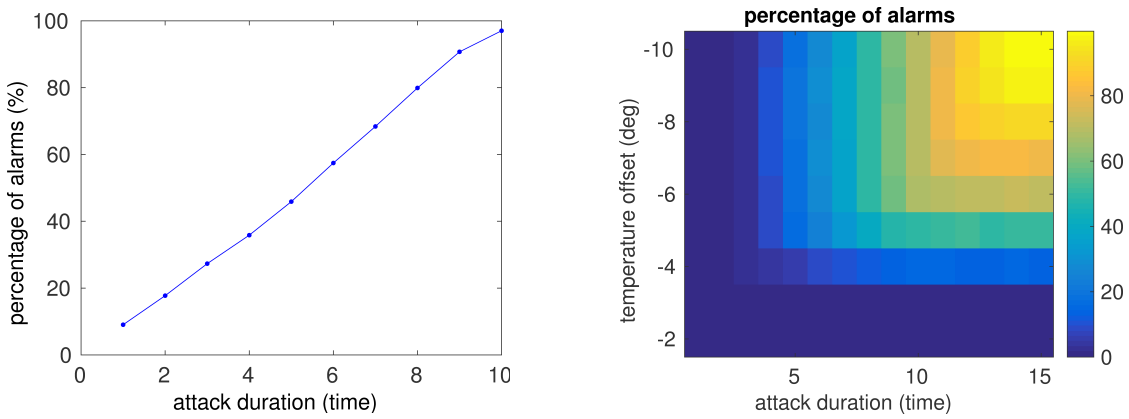

Fig. 6. A quantitative analysis of two different attacks.

we relied on the trace semantics of CCPSA to assess *attack tolerance/vulnerability* with respect to a given attack. Along the lines of GNDC [7], we defined a notion of *top attacker*, $Top(C)$, of a given class of attacks C, which has been used to provide sufficient criteria to prove attack tolerance/vulnerability to all attacks of class C (and weaker ones). As we remarked above, we believe that we could switch to a (bi)simulation semantics, preserved by parallel composition, which would in turn allow us to scale Theorem 1 and, more generally, our approach to bigger systems, but this will require a thorough investigation. Finally, we have provided a metric to estimate the impact of a successful attack on a CPS together with possible quantifications of the success chances of an attack. We proved that the impact of the most powerful attack $Top(C)$ represents an upper bound for the impact of any attack A of class C (and weaker ones).

Related work: A number of approaches have been proposed for modelling CPSs using *hybrid process algebras* [5], [2], [24], [8]. CCPSA shares some similarities with the ϕ-calculus [24]. However, unlike CCPSA, in the ϕ-calculus, given a hybrid system (E, P), the process P can dynamically change the evolution law in E. Furthermore, the ϕ-calculus does not have a representation of physical devices and measurement law, which are instead crucial for us to model cyber-physical attacks that operate in a timely fashion on sensors and actuators.

Among the 118 papers discussed in the comprehensive survey [31], 50 adopt a discrete notion of time similar to ours, 13 a continuous one, 48 a quasi-static time model, and the rest use a hybrid time model. Most of these papers investigate attacks on CPSs and their protection by relying on *simulation test systems* to validate the results.

We focus on the papers that are most closely related to our work. Huang et al. [12] were among the first to propose *threat models* for CPSs. Along with [13], [14], they stressed the role played by timing parameters on integrity and DoS attacks. Alternative threat models are discussed in [9], [10], [26]. In particular, Gollmann et al. [10] discussed possible goals (*equipment damage*, *production damage*, *compliance violation*) and *stages* (*access*, *discovery*, *control*, *damage*, *cleanup*) of cyber-physical attacks. In the current paper, we focused on the damage stage, where the attacker already has a rough idea of the plant and the control architecture of the target CPS.

A few works use formal methods for CPS security, although they apply methods, and most of the time have goals, that are quite different from ours.

Burmester et al. [3] employed *hybrid timed automata* to give a threat framework based on the traditional Byzantine faults model for crypto-security. However, as remarked in [26], cyber-physical attacks and faults have inherently distinct characteristics. Faults are considered as physical events that affect the system behaviour, where simultaneous events don't act in a coordinated way; cyber-attacks may be performed over a significant number of attack points and in a coordinated way.

In [28], Vigo presented an attack scenario that addresses some of the peculiarities of a cyber-physical adversary, and discussed how this scenario relates to other attack models popular in the security protocol literature. Then, in [29], [30] Vigo et al. proposed an untimed calculus of broadcasting processes equipped with notions of failed and unwanted communication. These works differ quite considerably from ours, e.g., they focus on DoS attacks without taking into consideration timing aspects or impact of the attack.

Cómbita et al. [4] and Zhu and Basar [32] applied *game theory* to capture the conflict of goals between an attacker who seeks to maximise the damage inflicted to a CPS's security and a defender who aims to minimise it [19].

Finally, there are three recent papers that were developed in parallel to ours: [20], [23], [22]. Rocchetto and Tippenhaur [23] introduced a taxonomy of the diverse attacker models proposed for CPS security and outline requirements for generalised attacker models; in [22], they then proposed an extended Dolev-Yao attacker model suitable for CPSs. In their approach, physical layer interactions are modelled as abstract interactions between logical components to support reasoning on the physical-layer security of CPSs. This is done by introducing additional orthogonal channels. Time is not represented.

Nigam et al. [20] work around the notion of Timed Dolev-

Yao Intruder Models for Cyber-Physical Security Protocols by bounding the number of intruders required for the automated verification of such protocols. Following a tradition in security protocol analysis, they provide an answer to the question: How many intruders are enough for verification and where should they be placed? They also extend the strand space model to CPS protocols by allowing for the symbolic representation of time, so that they can use the tool Maude [21] along with SMT support. Their notion of time is however different from ours, as they focus on the time a message needs to travel from an agent to another. The paper does not mention physical devices, such as sensors and/or actuators.

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 cyber-physical attackers and attacks, e.g., periodic attacks. 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 preliminary quantitative analysis we have given here by developing a suitable behavioural theory ensuring that our trace semantics considers also the probability of a trace to actually occur. We expect that *weak simulation quasimetrics* [17] will be useful to that extent.

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., [1]), 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.

Acknowledgements: We thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] 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.

[2] J. A. Bergstra and C. A. Middleburg. Process algebra for hybrid systems. *Thoretical Computer Science*, 335(2-3):215–280, 2005.

[3] M. Burmester, E. Magkos, and V. Chrissikopoulos. Modeling security in cyber-physical systems. *IJCIP*, 5(3-4):118–126, 2012.

[4] 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.

[5] P. Cuijpers and M. Reniers. Hybrid process algebra. *The Journal of Logic and Algebraic Programming*, 62(2):191–245, 2005.

[6] N. Falliere, L. Murchu, and E. Chien. W32.Stuxnet Dossier, 2011.

[7] 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.

[8] V. Galpin, L. Bortolussi, and J. Hillston. HYPE: Hybrid modelling by composition of flows. *Formal Asp. Comput.*, 25(4):503–541, 2013.

[9] 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.

[10] 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.

[11] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, 1995.

[12] 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.

[13] 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.

[14] 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.

[15] R. Lanotte and M. Merro. A Calculus of Cyber-Physical Systems. In *LATA*, volume 10168 of *LNCS*, pages 115–127. Springer, 2017.

[16] R. Lanotte, M. Merro, R. Muradore, and L. Viganò. A Formal Approach to Cyber-Physical Attacks. *CoRR*, abs/1611.01377, 2016.

[17] R. Lanotte, M. Merro, and S. Tini. Weak simulation quasimetric in a gossip scenario. In *FORTE*, volume 10321 of *LNCS*. Springer, 2017.

[18] D. Macedonio and M. Merro. A semantic analysis of key management protocols for wireless sensor networks. *Science of Computer Programming*, 81:53–78, 2014.

[19] 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.

[20] V. Nigam, C. Talcott, and A. A. Urquiza. Towards the Automated Verification of Cyber-Physical Security Protocols: Bounding the Number of Timed Intruders. In *ESORICS*, volume 9879 of *LNCS*, pages 450–470. Springer, 2016.

[21] P. C. Ölveczky and J. Meseguer. Semantics and pragmatics of real-time maude. *Higher-Order and Symbolic Computation*, 20(1-2):161–196, 2007.

[22] M. Rocchetto and N. O. Tippenhauer. CPDY: Extending the Dolev-Yao Attacker with Physical-Layer Interactions. In *ICFEM*, volume 10009 of *LNCS*, pages 175–192, 2016.

[23] M. Rocchetto and N. O. Tippenhauer. On Attacker Models and Profiles for Cyber-Physical Systems. In *ESORICS*, volume 9879 of *LNCS*, pages 427–449. Springer, 2016.

[24] W. C. Rounds and H. Song. The ϕ-calculus: A language for distributed control of reconfigurable embedded systems. In *HSCC*, volume 2623 of *LNCS*, pages 435–449. Springer, 2003.

[25] J. Slay and M. Miller. Lessons Learned from the Maroochy Water Breach. In *Critical Infrastructure Protection*, volume 253 of *IFIP*, pages 73–82. Springer, 2007.

[26] A. Teixeira, I. Shames, J. Sandberg, and K. H. Johansson. A secure control framework for resource-limited adversaries. *Automatica*, 51:135–148, 2015.

[27] U.S. Chemical Safety and Hazard Investigation Board, T2 Laboratories Inc. Reactive Chemical Explosion: Final Investigation Report. Report No. 2008-3-I-FL, 2009.

[28] R. Vigo. The Cyber-Physical Attacker. In *SAFECOMP*, volume 7613 of *LNCS*, pages 347–356. Springer, 2012.

[29] R. Vigo. *Availability by Design: A Complementary Approach to Denial-of-Service*. PhD thesis, Danish Technical University, 2015.

[30] R. Vigo, F. Nielson, and H. Riis Nielson. Broadcast, denial-of-service, and secure communication. In *IFM*, volume 7940 of *LNCS*, pages 412–427. Springer, 2013.

[31] Y. Zacchia Lun, A. D'Innocenzo, I. Malavolta, and M. D. Di Benedetto. Cyber-Physical Systems Security: a Systematic Mapping Study. *CoRR*, abs/1605.09641, 2016.

[32] Q. Zhu and T. Basar. Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: games-in-games principle for optimal cross-layer resilient control systems. *IEEE Control Systems*, 35(1):46–65, 2015.