Assessing the Severity of Smart Attacks in Industrial Cyber Physical Systems

Abdelaziz Khaled, Samir Ouchani, Zahir Tari, Khalil Drira

Assessing the Severity of Smart Attacks in Industrial Cyber Physical Systems

Abdelaziz Khaled*[1], Samir Ouchani†[2], Zahir Tari‡[3], and Khalil Drira§[4]

[1]Evina, 22 Rue Chapon, Paris
[2]LINEACT CESI, Aix-en-Provence, France
[3]School of Science, RMIT University, Melbourne, Australia
[4]CNRS, LAAS, Toulouse, France

Abstract

Industrial cyber-physical systems (ICPS) are heterogeneous inter-operating parts that can be physical, technical, networking, and even social like agent operators. Incrementally, they perform a central role in critical and industrial infrastructures, governmental, and personal daily life. Especially with the Industry 4.0 revolution, they became more dependent on the connectivity by supporting novel communication and distance control functionalities, which expand their attack surfaces that result in a high risk for cyber-attacks. Furthermore, regarding physical and social constraints, they may push up new classes of security breaches that might result n serious economic damages. Thus, designing a secure ICPS is a complex task since this needs to guarantee security and harmonize the functionalities between the various parts that interact with different technologies. This paper highlights the significance of cyber-security infrastructure and shows how to evaluate, prevent, and mitigate ICPS-based cyber-attacks. We carried out this objective by establishing an adequate semantics for ICPS's entities and their composition, which includes social actors that act differently than mobile robots and automated processes. This paper also provides the feasible attacks generated by a reinforcement learning mechanism based on multiple criteria that selects both appropriate actions for each ICPS component and the possible countermeasures for mitigation. To efficiently analyze ICPS's security, we proposed a model checking based framework that relies on a set of predefined attacks from where the security requirements are used to assess how well the model is secure. Finally, to show the effectiveness of the proposed solution, we model, analyze, and evaluate the ICPS security on two real use cases.

*Corresponding Author: abdelaziz.khaled@evina.fr
†souchani@cesi.fr
‡zahir.tari@rmit.edu.au
§khalil@laas.fr

Keywords. CPS, Smart Attacks, Cyber-Security, Formal methods, Artificial intelligence, Knowledge, Probabilistic Verification, Countermeasures, Reinforcement Learning, Multi-criteria Selection.

1 Introduction

With the immense connectivity and full automation provided by Industry 4.0, we witness an exponential expansion in the development and deployment of Cyber-Physical Systems (CPS) [3] in various critical domains. From the existing variety of CPS, we focus mainly on Industrial Cyber-Physical Systems (ICPS) that are an integration of computation, networking, industrial and physical processes, and social actors as agents. Nowadays, ICPS are playing progressively an important role in more critical infrastructure, governmental, and everyday life system (e.g. water distribution system, health-care systems, electrical power grids, oil and natural gas distribution, household appliances, transportation systems, etc.) [18, 3]. Theoretically, they are expected to be immune against all cyber-attacks and to be free of vulnerabilities, which is practically difficult in daily real-life systems.

To build a more efficient, stable and robust ICPS, many features like computation, communication, and control are integrated while security is not appropriately tackled [17, 4]; especially one of the fundamental issues is the heterogeneity between ICPS entities while they have a variation of components with different aspects connected in many ways. For example, hardware components can be sensors, embedded systems, and actuators. Further, they might cover also different chain tools of software for monitoring and control. Or either physical components like motors, tracks, tanks, etc. By relying on attack surface that is defined as a subset of resources that an attacker can exploit [13], each component and its integration might increase the attack surface of an ICPS; and consequently contributing to the success of such an attack. Understanding the current ICPS security trends, including vulnerabilities and weaknesses, attacks and mitigation mechanisms could provide a deeper understanding of the existing security disposition and challenges of ICPSs. ICPSs are generally distributed across a wide range of distant geographic areas, they therefore collect a huge amount of non-formatted messages and data for analysis and decision making. Based on the collected information, the decisions rely generally on sophisticated machine learning (ML) techniques, which are known to consume more (execution) time and memory. Accordingly, breaches in the information collection step could harm a wide-scale data of distinct sensitivity level, and these can occur during the different stages of the system's operations (e.g. data collection and extraction, transmission and processing, storage).

Unfortunately, most of the existing ICPS design methodologies do not consider data protection, especially the complexity of ICPSs and the heterogeneity of its components complicate security hardening and privacy protection [17, 4]. In particular, with the complexity of ICPS design, threats and vulnerabilities become more difficult to detect/assess/etc as the ICPS environment is highly dynamic and new security threats may stem easily. Additionally, it is also hard to check, identify, and trace attacks, which may originate from one or more sources. One should be able to point out

ICPS's loopholes that make them vulnerable to different attacks and therefor devise appropriate methods to defend against them. Further, we believe that the evaluation of ICPS's security throughout their life cycle is a necessary step to guarantee their security when deployed.

This work describes the SSA-ICPS framework[1] which can be used to assess the severity of attacks in industrial cyber physical systems (ICPS) by covering the following three parts: (1) modeling, (2) analysis, and (3) mitigation. First, the modeling part implements a flexible and extensible library of a predefined templates of the ICPS architectures and components, as well as a variety of the used communication protocols. In addition, it models smart attacks retailed to each ICPS's component. These attacks are powerful and smart as they rely on reinforcement learning that maximizes the reward (with an optimal policy search) as well as selects appropriate actions given a set of criteria. Further, SSA-ICPS models ICPS's behaviour where a developer selects a proper ICPS architecture and later refines its components behaviour following a well-guided predefined syntax. The library covers a set of predefined security requirements, which are needed to be satisfied on ICPS as well as a set of countermeasures, to recover or reinforce the system when needed. The analysis part forms initially a malicious environment by instantiating then composing the potential attacks within the modeled ICPS. Then, it instantiates the security properties needed to be satisfied on the composed environment (ICPS/attacks). To overcome the analysis complexity, SSA-ICPS relies on the probabilistic model checking PRISM that helps to rapidly find the possible attack scenarios as well as to measure the severity of the succeeded ones. Finally, the mitigation procedure reinforces the ICPS model using a recommendation process that selects the appropriate countermeasures configuration to be deployed for the identified/detected attacks.

The main contributions of this paper can be summarised as follows.

- ICPS architecture that defines the main components of ICPS by covering: the social aspect of actors, hardware (sensors, etc), software (applications, web services, etc), and physical components (infrastructures, motors, etc). Also, the architecture defines the way of how each component communicates, interacts, and is composed with others.
- ICPS semantics that models the architecture in a process algebra formalism by capturing the underlying of ICPS components, their behaviours, as well as their composition operators.
- The ICPS requirements that express in Probabilistic Computation Tree Logic (PCTL) formula the set of functional and security properties needed to check and reinforce ICPS.
- A security analysis technique based on probabilistic verification that transforms the ICPS model into the PRISM input language.
- A library of attacks related to the different aspects of components carried by the ICPS architecture. Each attack maximizes its rewards by using reinforcement

[1] SSA stands for "Severity of Smart Attacks in Industrial Cyber Physical Systems."

learning and selects appropriate actions through the use of a multi selection criteria algorithm.

- A library of countermeasures that match the different attack scenarios.
- A mitigation mechanism that recovers and reinforces the modeled ICPS from attacks.

The next section surveys existing solutions, and Section 3 provides details of the proposed security assessment framework. Section 4 discusses the experimental results, and finally we conclude this paper in Section 5.

2 Related work

This section reviews and discusses approaches that deal with modeling, functional analysis, and security specification techniques, as well as the communication protocols for ICPS.

Cheh *et al.* [6] analyzed the safety of railway systems corresponding to different classes of venomous actors based on to their abilities defined in the system access control. Initially, they constructed a hybrid automata that models a railway system, where each attack capability is defined as a pattern that represents its effects when appended to a component in the system. This combination describes the system behaviour and the possible actions of an attacker that might be executed. Then, they used a statistical model checking to assess the safety of the system for various input configurations of railways. Chen *et al.* [7] studied the detectability of data deception in CPS by assuming that an attack detector has access to a linear function of the initial system state and it cannot be changed by an attacker. In this context, the attack has the power to be undetectable by any dynamic attack detector under any specific constraints. The attacks are characterized to be maintained arbitrarily for long periods without being detected. Initially, they defined the zero state by inducing only the attacks that remain dynamically undetectable regardless the available information to the attack detector at the initial state. Cheh *et al.* [6] and Chen *et al.* [7] consider attacks that are initiated when the system starts the execution without showing the impact of each attack action on the behaviour of the system functionality. Further, they do not consider how to overcome them.

Bakirtzis *et al.* [5] proposed to evaluate the security level of CPS at each step of their life cycle development with the focus on deploying and operating safety-critical applications. To allow the analysis of vulnerabilities before deployment, they defined a taxonomy of attributes, which is a generalized schema that can capture mainly the characteristics of a safety-critical application. Then, they proposed to cover the maximum space of attack vectors associated with the model. Thus, the possible attack vectors are matched in order to mitigate them at the design level using SysML. Linzini *et al.* [9] proposed a framework to check and evaluate attacks in socio-technical systems. The model supports probability and cost, where the intruder model can challenge honest agents. For analysis, they express the security properties using PCTL and PRISM to

validate the correctness of the properties on the model. Ouchani et Linzini [13] formalized socio-technical physical systems as SysML activity diagrams and provided a library of attacks taken from standard catalogs of social engineering and technical attacks (CAPEC). Then, each diagram is analyzed separately by computing the probability of the weakest points that might host attacks. Bakirtzis *et al.* [5], Linzini *et al.* [9], and Ouchani et Linzini [13] rely on a taxonomy of attacks without showing how to instantiate an attack pattern for a specific use case. In addition, they focus more on matching the countermeasures within attacks instead of measuring the severity of attacks then select the appropriate countermeasure.

Mitchell and Chen [11] used stochastic Petri nets to develop an analytical model that can capture the interactions between the adversary and the defense models of CPS by focusing on three failures: attrition, perversion, and exfiltration. Agrawal *et al.* [2] try to design more secure CPS by relying on two categories of attacks. The attacks from outside were considered as a threat model, which might not be able to connect to an insider threat with the physical access to a CPS. At the same time, they included an attack detection mechanism for an insider attack with a physical access to a CPS. Thus, the dynamics of the system are exploited to detect adversaries based on the laws of Physics. Rocchetto *et al.* [15] proposed to extend Dolev-Yao's attacker model in order to analyze the security of CPS. A set of rules are used to define potential actions of attackers with respect to messages exchanged between parties during a protocol execution. Also, it allowed additional orthogonal interaction channels between parties. Physical properties such as locations and distances were included in the rule set. The previous approaches focus more on attacks related to physical properties instead of looking how to exploit technical vulnerabilities to command the physical processes remotely.

Rocchetto *et al.* [16] proposed a class of attacks that can retrieve, update, actuate, and command physical processes. The physical layer is designed to include attack vectors and their associated mitigation mechanisms that cover the attackers' capabilities by relying on a taxonomy that classifies and compares attacker actions. Finally, they select a set of attacker profiles that they consider them more powerful. Puys *et al.* [14] assessed the security of industrial systems by considering attackers that have already shown their ability to abuse some security breaches to earn access into the industrial systems. To find such attacks, they took into account different parameters especially the behaviour of the process, the safety properties to be ensured, and the possible positions and abilities of attackers. Both contributions try to find potential attacks from a predefined classes without a mitigation process. Further, they do not estimate the sevirity of each attack.

Xiaoxue *et al.* [10] described the existing threats that are dependent to the security and safety of CPS that are modeled by using a hierarchical method of division and integration. They proposed a quantitative analysis and modeling approach to express threat propagation using a probabilistic colored Petri net model that includes basic models, rules, logical operators and transitions as a menace breeding between nodes. The weights of the connections in the attack model are computed using a mixed-strategy attack-defense game approach by solving the Nash equilibrium. Adepu and Mathur [1] designed attacker models dedicated to CPS by generating a parameterized attack models in order to capture both physical and cyber attack abilities. For each class of attacks,

they look for the common possible parameters between cyber and physical attributes. Ouchani [12] proposed to analyze the functional correctness of IoT in a mixed environment. A formal model is proposed to cover various aspects such objects and services. For the analysis part, the approach relied on the PRISM probabilistic model checking by transforming an IoT model into PRISM source code, and expressing the functional properties in PCTL. The model covered the main components of IoT systems, however the framework suffers from the model checking inherited limitations and the security properties are not specified. Those contributions presented a formal modeling for CPS without showing how it has been developed. Further, the developed attacks are generic without relying on the provided models.

With respect to the reviewed approaches, the present work try to capture the underlying semantics of ICPS, and develop an automatic way to generate the behaviour of a concrete model. Further, it develops an extensible library that can be updated for attacks and countermeasures. Furthermore, it relies on PRISM model checker to assess the security of ICPS.

3 The security assessment framework

The SSA-ICPS framework depicted in Figure 1 assesses the severity degree of attacks in industrial cyber physical systems (ICPS) by covering the modeling, the security analysis, attacks, and mitigation of ICPS. For the modeling part, SSA-ICPS develops a predefined library of flexible and extensible templates for the ICPS architectures, components, and communication protocols. Further, it models different smart attacks related to every ICPS component. To precisely describe an ICPS's behaviour, a user has only to rely on the library to design a complete ICPS as well as to specify the behaviour of each component following a well-guided predefined syntax. Additionally, it covers the set of security requirements needed to be satisfied as well as a set of countermeasures to recover and reinforce the system under development. The analysis module instantiates first the possible attacks specific to the modeled ICPS so to compose them together to form a malicious environment. Also, it instantiates the security properties to check if they are valid or not within this composition. To overcome the analysis complexity, SSA-ICPS has an acceleration procedure that makes the checking process much easier and helps to rapidly find all possible attack scenarios as well as to measure the severity of each succeeded one. Finally, to reinforce the ICPS model, the recommendation process produces the possible optimal countermeasures configuration for the attacks.

3.1 System's model

This section describes the formal model that takes into consideration the ICPS architecture previously discussed (see Figure 1) as a composition of interconnected nodes: physical objects (devices and controllers, e.g. sensors and buildings), mobiles applications, cloud and computing online services, and social agents (people).

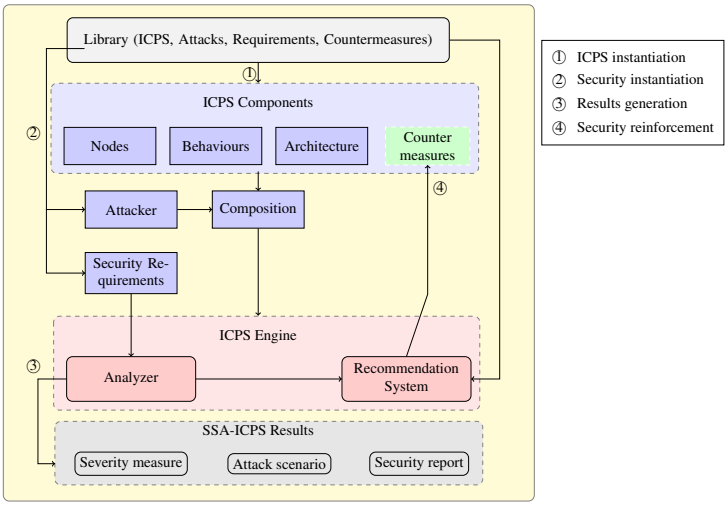

Figure 1: SSA-ICPS framework.

3.1.1 Node

This is the main entity describing an ICPS, and it is defined as tuple $\langle id, attr, Actuator, \Sigma, Beh \rangle$, where:

- id is a fixed set of insignia $id_\epsilon, \cdots, id_i, \cdots \in id$ identifying nodes where id_ϵ is the empty node.
- $attr : id \to 2^{\mathcal{T}}$ evaluates the attributes of a node, such that $\mathcal{T} = \{p, c, m, d, r\}$ where p, c, m, d, and r stand respectively for physical, container, movable, destroyable, and reproducible.
- $Actuator : id \to loc \times 2^{id} \times id \times \mathbb{B}$ is a function that evaluates the tuple $\langle loc, cont, key, locked \rangle$ specifying the status of a node id_i by returning respectively its: location, contained objects, key, and if it is locked or not.
- Σ is a finite set of atomic actions that a given node can execute, where:

$$\begin{aligned}\Sigma = \{&\texttt{Start}, \texttt{Terminate}, \texttt{Send}(id_i, id_j), \texttt{Receive}(id_i, id_j),\\ &\texttt{Update}(id_i, id_j), \texttt{Lock}(id_i, id_j), \texttt{Unlock}(id_i, id_j),\\ &\texttt{Move}(l, l') : id_i, id_j \in id \text{ and } l, l' \in loc\}.\end{aligned}$$

  Start and Terminate starts and terminates the process of a node, $\texttt{Send}(id_i, id_j)$ and $\texttt{Receive}(id_i, id_j)$ sends and receives id_i to/from id_j, $\texttt{Update}(id_i, id_j)$ updates id_i by id_j, $\texttt{Lock}(id_i, id_j)$ and $\texttt{Unlock}(id_i, id_j)$ lock and unlock id_i with id_j, respectively.
- $Beh : id \to \mathcal{L}$ returns the expression written in the language $\mathcal{L}$ that describes the behaviour of a node. The syntax of $\mathcal{L}$ is given by:

B`Start` |`Terminate` | $\alpha \cdot B$ | $\alpha \cdot B +_g \alpha \cdot B$ | $\alpha \cdot B +_p \alpha \cdot B$ | $\alpha \cdot B + \alpha \cdot B$ | $\alpha \cdot B|_\alpha \alpha \cdot B$ | $\alpha \cdot B || \alpha \cdot B$, where $\alpha \in \Sigma$ and " $\cdot$ " composes sequentially the actions, and $+_g$ is a guarded choice decision, $+_p$ is a probabilistic decision, $+$ is a non deterministic choice, $|_\alpha$ is a synchronization on the action α, and $||$ is the interleaving operation.

3.1.2 Node's behaviour

To characterize the behaviour of a node, say n, we first define its state and how this can change as a labeled state transition system $\langle S, s_0, \rightarrow \rangle$ where S is all possible states of n, $s_0 \in S$ is its initial state, and $\rightarrow \subseteq \ (S \times L \times \mu \times S)$ is the closure transition relation between states labeled by $l \in L$ and executed with a probability $p \in \mu$ (μ is a probabilistic distribution). A transition $\rightarrow$ defines the changes of a node n and represented by a set of operational semantics rules. For example, the SRT rule describes the initial execution of a node n.

$$\frac{B_n = \texttt{Start}.B_n^{'}}{\langle id_n, -, \langle -, - \rangle, - \rangle \xrightarrow{start_n} \langle id_n, B_n^{'}, \langle -, - \rangle, - \rangle} \text{ SRT}$$

The rule UP updates sending a value $[\![m]\!]$ of the variable x by a value $[\![m']\!]$ in a node n.

$$\frac{B_n = \texttt{Update}(x, [\![x']\!]).B_n^{'} \wedge x \in cont(n)}{\langle id_n, -, < -, \{x, [\![x]\!]\} >, - \rangle \xrightarrow{up_x} \langle id_n, B_n^{'}, < -, \{x, [\![x']\!]\} >, - \rangle} \text{ UP}$$

The rule SYN describes a node n sending a value $[\![m]\!]$ of a variable x to another node n' carried by the variable y.

$$\frac{\begin{array}{c} B_n = \texttt{Send}(n', x).B_n^{'} \wedge x \in cont(n) \wedge [\![x]\!] = m \neq \epsilon_o \\ B_n = \texttt{Receive}(n, y).B_{n'}^{'} \end{array}}{\begin{array}{c} \langle \langle id_n, -, < -, \{x, [\![x]\!]\} >, - \rangle, \langle id_{n'}, -, < -, \{y, -\} >, - \rangle \rangle \xrightarrow{sndrcv_x} \\ \langle \langle id_n, B_n^{'}, < -, \{x, [\![x]\!]\} >, - \rangle, \langle id_{n'}, B_{n'}^{'}, < -, \{y, [\![x]\!]\} >, - \rangle \rangle \end{array}} \text{ SYN}$$

The set of all operational semantics rules are given in Appendix 6.1.

3.1.3 `ICPS` Architecture

The `ICPS` architecture is the main structure carrying a `ICPS` system. In harmony, it regroups all nodes together. It is modeled by a directed graph where the vertices are the set of nodes and edges are their interaction. Formally, the `ICPS` architecture is a tuple $\langle V, E, \pi, CM \rangle$, where:

- $V = N$ is a finite set of nodes: $n_0, \cdots$.
- $E \subseteq N \times N$ is a finite set of links between nodes, such that for every $e \in E$ there is a pair of channels $e_{ij} = \langle n_i, n_j \rangle$ and $e_{ji} = \langle n_j, n_i \rangle$.

- $\pi : E \longrightarrow Prot$ returns the communication protocol for an edge in E.
- $CM : N \cup E \longrightarrow C$ returns the proper countermeasure to a node or an edge.

The communication protocol synchronizes and organizes the communication and the interactions between nodes. However, we have also defined many communication protocols depends on the nature of each node. Formally, $prot \in Prot$ is the tuple $\langle Prot_{o,s}, Prot_{s,s}, Prot_{o,o}, Prot_{h,o} \rangle$ where $Prot_{o,s}$ ensures the communication between objects and services, $Prot_{s,s}$ between services, $Prot_{o,o}$ between objects, $Prot_{h,o}$ between social actors and abjects.

For $Prot_{s,s}$, we would take the example of Modbus protocol described with a set of action and events Σ', where we have: $\Sigma' = \{\textit{start}_{\mathtt{L}}(m), \textit{send}_{\mathtt{L}}(a_1, a_2, m_1), \textit{recieve}_{\mathtt{L}}(a_2, a_1, m_2)\}$. The action $\textit{start}_{\mathtt{L}}(m)$ is the initialisation of the protocol, $\textit{send}_{\mathtt{L}}(a_1, a_2, m_1)$ denotes sending the message m_1 from agent a_1 to a_2, $\textit{recieve}_{\mathtt{L}}(a_2, a_1, m_2)$ means receiving m_2 by a_2 from a_1. The label $\mathtt{L}$ gives the order of events. Further, we consider the execution of the behaviour as a *session*, and formally it is defined as a tuple $s = \langle id, \sigma, \varphi, \varepsilon \rangle$, where id is the identifier of the session, σ are the initiator and the responder, φ are the values of the variables, ε is a set of events.

Let $prot \in Prot$ be the specification of a protocol and $\mathcal{A}$ is the set of active sessions. The basic operational semantics rule of $Prot$ are given in Table 1. The `Start` rule states that a new session can only be created if its identifier has not been used yet. The $send$ rule shows that if the session executes the $send$ event then the message will be added to the buffer $\mathcal{B}_s$ and executes the next event ε. The $receive$ rule expresses that there is a session where the next event is $receive$ and $\mathcal{B}_r$ contains a message. The latter will be removed from the buffer $\mathcal{B}_r$ to proceed the execution of the next event ε.

$$\frac{s = (id, \sigma, \varphi, \varepsilon) \in Prot \;\; id \notin \mathcal{A}}{\langle \mathcal{B}_s, \mathcal{B}_r, \mathcal{A} \rangle \xrightarrow{\text{start(s)}} \langle \mathcal{B}_s, \mathcal{B}_r, \mathcal{A} \cup \{s\} \rangle} \; start$$

$$\frac{e = \textit{send}_{\mathcal{L}}(a_1, a_2, m_1) \;\; (id, \sigma, \varphi, [e].\varepsilon) \in \mathcal{A}}{\langle \mathcal{B}_s, \mathcal{B}_r, \mathcal{A} \rangle \xrightarrow{\text{send(s)}} \langle \mathcal{B}_s \backslash \{m\}, \mathcal{B}_r \bigcup \{m\}, \mathcal{A} \rangle} \; send$$

$$\frac{e = \textit{recieve}_{\mathcal{L}}(a_2, a_1, m_2) \; (id, \sigma, \varphi, [e].\varepsilon) \in \mathcal{A} \; m \in \mathcal{B}_r}{\langle \mathcal{B}_s, \mathcal{B}_r, \mathcal{A} \rangle \xrightarrow{\text{recieve(s)}} \langle \mathcal{B}_s \backslash \{m\}, \mathcal{B}_r \bigcup \{m\}, \mathcal{A} \rangle} \; recieve$$

Table 1: A session operational semantics rules.

We defined the operational semantics rules using a labelled transition system $\langle S, S_0, \longrightarrow \rangle$, where S is the set of all states in the system, S_0 is the initial state and $\longrightarrow \subseteq (S \times L \times S)$ is the transition relation between states labeled by L, where $L \in \mathtt{labels}(\Sigma')$ and `labels` is a function that returns a label for each action of Σ'.

Figure 2 depicts the architecture of an `ICPS` using `ICPS`-graph, where C is a multi-protocol client, S_1 and S_2 are Modbus and OPC-UA servers, P is a pro-

grammable logic controller (PLC), Se is a sensor, V is a valve, and M is a motor. All nodes are connected through digital (e.g. Modbus, OPC-UA) and analogic channels.

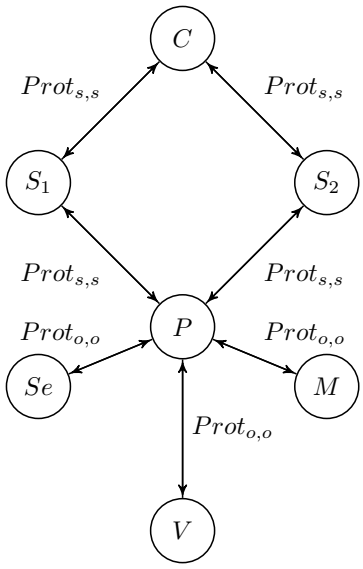

Figure 2: ICPS Graph.

3.1.4 Countermeasures

To harden a system against various attacks, we use a suitable security measure. A countermeasure could take different forms, such as recommendations about how to modify the architecture of the system, how to assure the security of the system or a software, and/or how to add or modify physical measures ($e.g.$ door digital code, firewall, etc.). Formally a countermeasure $c \in C$ is a tuple $\langle id, N, \Sigma_{cm}, Beh_{cm} \rangle$, where:

- id is a finite set of tags $id_1, \cdots, id_i, \cdots, id_n$ identifying a software or physical measures.
- N is a finite set of nodes.
- Σ_{cm} is a set of actions can be executed by a countermeasure cm, where:

$$\begin{aligned}\Sigma_{cm} = \{&\texttt{Start}, \texttt{Terminate}, \texttt{Filters}(net_i, net_j),\\ &\texttt{Scan}(n_i), \texttt{Crypt}(n_i, n_j), \texttt{Decrypt}(n_i, n_j), \texttt{Auth}(n_i),\\ &\texttt{CheckPers}(e_i), \texttt{Lock}(n_i, d_j), \texttt{Unlock}(n_i, d_j) :\\ &net_i, net_j \in Net, n_i, n_j \in Node, e_i \in E \text{ and } k_j \in Key\}\end{aligned}$$

  `Start` and `Terminate` starts and terminates the process of a countermeasure, $\texttt{Filters}(net_i, net_j)$ filters and controls incoming and outgoing network traffic between net_i and net_j based on predefined security rules, $\texttt{Scan}(n_i)$ scans

the node n_i ($e.g.$ $computer$) from malwares, $\texttt{Crypt}(n_i, n_j)$ encrypts the communication between the nodes n_i and n_j, $\texttt{Decrypt}(n_i, n_j)$ decrypts the communication between nodes n_i and n_j, $\texttt{Auth}(n_i)$ uses an authentication method ($e.g.$ $password$) for the node n_i, $\texttt{CheckPers}(a_i)$ exams the personalty of an employee e_i, $\texttt{Lock}(n_i, k_j)$ and $\texttt{Unlock}(n_i, k_j)$ locks and unlocks n_i with n_j, respectively.

- Beh_{cm}: $CM \longrightarrow \mathscr{L}_{cm}$ returns the behaviour of a countermeasure. The syntax of $\mathscr{L}_{cm}$ is given by: $B\texttt{Start} \cdot B \mid \texttt{Terminate} \mid \alpha \cdot B \mid \alpha \cdot B +_g \alpha \cdot B$, where α is an action and " $\cdot$ " composes sequentially the actions, and $+_g$ is a guarded choice decision.

3.2 Attacker model

Based on the literature, Dolev-Yao attacker model [8] is the most powerful one that can access and manipulate arbitrarily all the network traffic. This attacker model is usually employed for the identification attacks (e.g, Web Application). It can intercept messages and analyze them if he possesses the corresponding keys for decryption. Also, it can generate messages from his knowledge, and send them as any honest or impersonate agent.

However, this attacker model suffers from the state explosion and affects only the network layer as a main-in-the-middle. Our proposed attacker model is an improved version using the reinforcement learning and the multi-criteria analysis, where the attacker can perform all type of attacks (i.e. network, physical, software, and social engineering). In addition, we avoid the state explosion problem by relying on the reinforcement learning based on the multi-criteria analysis that makes the decision more deterministic and smart.

Hence, we describes the potential attacks proper to ICPS components and their interactions. As shown in Figure 3, the ICPS attacker model has two components: 1) *Data* contains knowledge, personality and skills, and 2) the *Analyzer* that is an inference based engine generating attacks according to data, the received inputs, and its skills and inference levels. The attacker model could be one from the internal employer who works inside or outside (e.g. disgruntled employees or a social engineering victims), a malicious software, or any node with powerful capabilities and techniques.

The ICPS attacker depicted in Figure 3 takes as input the channel $\boldsymbol{Chan}$, the physical access $\boldsymbol{Phy}$, and the human interaction $\boldsymbol{Hum}$. $\boldsymbol{Chan}$ means that the attacker intercepts the message between the ICPS components. $\boldsymbol{Phy}$ means that the attacker has physical access (e.g. open a door). $\boldsymbol{Hum}$ means that the attacker interacts or communicates with an employ from inside.

3.2.1 Data

The data $\boldsymbol{\omega}$ of the attacker is a tuple $\langle K, P, S \rangle$, where K represents its knowledge, P is the personality of the attacker, and S is the set of skills.

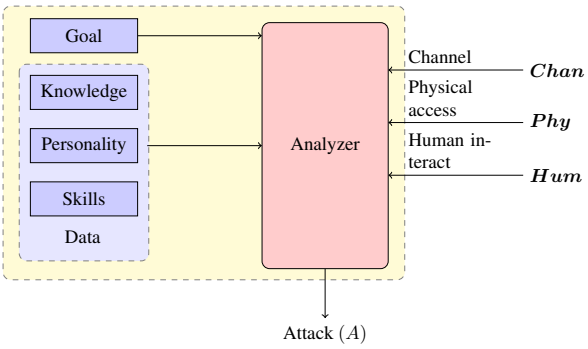

Figure 3: ICPS attacker model.

Knowledge

This is the core component of the attacker model. It contains the system model ($\mathcal{M}$), a secure information ($\mathcal{S}ec$), and algorithms of control or security techniques ($\mathcal{A}lg$). Formally, the knowledge of the attacker ($\mathcal{K}$) is a tuple, $\mathcal{K} = \langle \mathcal{M}, \mathcal{S}ec, \mathcal{A}lg \rangle$. Consequently, the knowledge in data can change according to the type or the level of attackers.

Personality

To fulfill the personality requirements, we rely on the well know theory from psychology called *the big five personality traits*, and also known as the five-factor model (FFM). Each factor represents a type of personality (Openness to experience, Conscientiousness, Extra-version, Agreeableness, Neuroticism) and the highest factor from the five above is the personality of the person. In our work, we are interested in the last factor Neuroticism, because the people who have a high level in this factor have emotions like: anger, anxiety, depression, and vulnerability. Formally, the personalty is a vector $\mathcal{P}er$, where each element is an emotion e in the state i.

$$\mathcal{P}er = \begin{vmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{vmatrix} \text{ where } \forall i \in [1, n], e_i = \begin{vmatrix} 1 & positive \\ 0 & absence \\ -1 & negative \end{vmatrix}$$

Goal

An attack is a set of actions or small attacks where each action has a goal, and formally a goal $\mathcal{G}$ is a set $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$. An attacker must achieve all goals where g_i is an expression specifying a state of nodes.

3.2.2 Analyzer

Since a given attack $\mathcal{A}$ is a set of actions, the choice of this set depends on the set of the selected criteria. We use Markov Decision Process (MDP) and a multi-criteria analysis to model the decision of the attacker as well as the reinforcement learning to select the optimal sequence of actions to maximize the attacker assets. Algorithm 1 describes how the attacker could choose an action based on its goal while minimizing the cost.

Algorithm 1 Generation attack actions.

Input: $\mathcal{M}_{mdp}$ //The system behaviour with a graph structure.
Output: π //The attack sequence of actions.
$\forall s \in S : U(s) = 0$;
$\lambda = 0.5$;
$update = 1$;
while $update > \epsilon$ **do** //Update the cost U to achieve s.
  for $(s \in S)$ **do**
    $U(s) = \mathcal{R}(s) + \lambda \max_{a \in A(s)} \sum_{s' \in S} \mathcal{P}(s', s, a) \times U(s')$; //Measure the cost of a state s.
    $update = \sum_{s' \in S} |U(s) - U(s')|$;
  end for
end while
$\pi(s) = arg \max_{a} \sum_{s' \in S} P(s', s, a) \times U(s')$; //Select the path maximizing the reward.

Algorithm 1 takes as input a Markov Decision Process model $\mathcal{M}_{mdp}$ to produce the set of actions π. $\mathcal{M}_{mdp}$ is a tuple $\mathcal{M}_{mdp} = (\mathcal{S}, \mathcal{A}, s_0, \mathcal{R}, \mathcal{P}, \gamma)$, where:

- $\mathcal{S}$ is a set of finite states $s_2, s_2, etc.$
- $\mathcal{A}$ is a set of finite actions $a_1, a_2, etc.$
- s_0 is the initial state.
- $\mathcal{R}(s)$ is a reward function that returns the utility for each state s.
- $\mathcal{P}(s, a, s')$ the probability of being in the state s' after executing the action a from state s.
- γ is a discount factor $0 < \gamma < 1$.

To achieve a state s_i in the model representing a sub-goal g_i, the attacker should execute an action a_i according to a sequence of decisions based on data. $\mathcal{R}(s)$ is

calculated with respect to the Weight Sum Method as follows

$$\mathcal{R}(s) = \sum_{j=1}^{m} w_j c_{ij},\ i = 1, 2, ..., n$$

whre c is a set of criterion to select an action and w denotes the relative weight of importance of c. The probability of a transition $\mathcal{P}(s, a, s')$ depends on the value of $\mathcal{R}(s')$ and the sum of all the $\mathcal{R}(s')$ for the successors of the state s. $\mathcal{P}(s, a, s') = \frac{\mathcal{R}(s')}{\sum_{\forall s' \in succ(s)} \mathcal{R}(s')}$

The selection of a transition depends of the utility of each state and the accumulated reward. In our case, the attacker will choose the actions to earn more rewards.

3.3 ICPS Engine

The ICPS engine contains two main parts. The first part is the analyzer that measures the severity of attacks and their scenarios. Further, it reports bugs and existing vulnerabilities. The second part relies on the security report to recommend the optimal countermeasures. It also provides hints the best practices for the developed ICPS system.

3.3.1 ICPS Analyzer

For the analysis, we rely on the probabilistic symbolic model checker PRISM that verifies probabilistic specifications over probabilistic models. A specification can be expressed either in the probabilistic computation tree logic (PCTL) and a model can be described using PRISM language.

A model can be a discrete-time Markov chains, continuous-time Markov chains, and Markov decision processes (MDPs). Alternatively, a model can be a probabilistic timed automata. PRISM also supports probabilistic automata. PRISM verification is efficient, since it stores models as binary decision diagrams and multi-terminal BDDs. To overcome the state explosion problem, PRISM has built-in symmetry reduction and implements some iterative numerical analysis like Jacobi and Gauss-Siedel.

In general, a given PRISM program is a composition of a set of *modules*. A module is evaluated over a fixed number of local variables, of type Boolean or integer. Whereas the state of a given module is formulated as the evaluation of its local variables, the global state of a PRISM program is the evaluation of all variables, local and global, for all modules. Further, the composition and the communication between PRISM modules adopt the operators developed by the CSP process algebra.

Basically a PRISM module defines the kernel behaviour of a PRISM program. At this end, the behaviour of a module is a collection of commands that can be probabilistic or Dirac. Textually, a probabilistic command is expressed by $[\alpha]\ g \rightarrow p_1 : u_1$+...+$p_m : u_m$, such as p_i are probabilities ($p_i \in]0, 1[$ and $\sum_{i=0}^{m} p_i = 1$), α is a label expressing the name of the action α, g is the guard represented as a propositional logic formula over all variables, local and global, and u_i describes the *update* (new value) for an

ensemble of variables. A given update expressed by $(v'_j = val_j)\&\cdots\&(v'_k = val_k)$, assigns only values val_i to local variables v_i. So, for a given action α, if the guard g is valid, then the update u_i is enabled with a probability p_i. In general, the guard is an expression consisting of the evaluation of all variables that are connected explicitly with the propositional logic operators. The Dirac case where $p = 1$ is a special case command expressed simply by: $[a]\ g \rightarrow u$.

Syntactically, a module named M is delimited by two keywords: the module head "`module` M", and the module termination "`endmodule`". Further, we can model costs with a reward module R delimited by keywords "`rewards` R" and "`endrewards`". A reward module is composed from a *state reward* or a *transition reward*. A state reward associates a cost (reward) of value r to any state satisfying g and it is expressed by $g : r$. A transition reward is specified by $[a]\ g : r$ to express that the transitions labeled a, from states satisfying g, are acquiring the reward of value r.

Finally for the analysis, a PRISM program P proper will be generated to the provided `ICPS` formalism. For that, we introduced the function $\mathcal{T}_P$ that assigns for each `ICPS` node behaviour its proper PRISM code fragment that is bounded by '`module node name`' and '`endmodule`' and the semantic rules of each action is expressed by a PRISM command.

For the semantic rule of any entity, its premises represent the guard of the entity PRISM command, whereas the update describes the consequence of the rule. For example, o_{o_2} is an atomic proposition showing the the object o possess o_2, l_a and l_o present the locations, and p_{o_3} precises the physicality attribute of o_3. The variables and propositions are evaluated first to describe the initial state of nodes by relying on the tuple obtained by the $Actuator$ proper to each entity. $\mathcal{T}_P$ implements this transformation for each entity depends its category, and here we consider the transformation of rules already presented in Section 3.1.

$$\mathcal{T}_P(\alpha) = \begin{cases} [Syn_{o_2}]o_{o_2} \wedge o_{1_{o_3}} \wedge \neg p_{o_2} \wedge \neg p_{o_3} \rightarrow (o'_2 = o_2); \\ [Syn_{o_2}]o_{o_2} \wedge o_{1_{o_3}} \wedge \neg p_{o_2} \wedge \neg p_{o_3} \rightarrow (o'_3 = o_2); \\ \textbf{iff:}\ \texttt{Send}(o_1, o_2) \in \Sigma_o^{o_1}, \texttt{Receive}(o_3, o_2) \in \Sigma_o^{o_2}. \\ \\ [Tak_{o_1}]l_a = l_o \wedge o_{o_2} \wedge \neg lock_o \wedge p_{o_2} \rightarrow (a'_{o_2} = \top); \\ [Tak_{o_1}]l_a = l_o \wedge o_{o_2} \wedge \neg lock_o \wedge p_{o_2} \rightarrow (o'_{o_2} = \bot); \\ \textbf{iff:}\ \texttt{Receive}(o, o_2) \in \Sigma^a. \\ \\ [loc_{o_1}]o_{o_1} \wedge o_{o_2} \wedge \neg k_{o_1} \wedge p_{o_1} = p_{o_2} \rightarrow (k'_{o_1} = \top); \\ [loc_{o_1}]o_{o_1} \wedge o_{o_2} \wedge \neg k_{o_1} \wedge p_{o_1} = p_{o_2} \rightarrow (o'_{o_1} = \top); \\ \textbf{iff:}\ \texttt{Lock}(o_1, o_2) \in \Sigma_o^o. \end{cases}$$

More details about the function $\mathcal{T}_P$ are provided in Appendix 6.2.

3.3.2 Security requirements

To specify `ICPS` security requirements and policies we use PCTL syntax presented as follows

$$\phi \ ::= \ \top \mid ap \mid \phi \wedge \phi \mid \neg\phi \mid \mathrm{P}_{\bowtie p}[\psi] \mid \mathrm{R}[\psi]$$
$$\psi \ ::= \mathrm{X}\phi \mid \phi \mathrm{U}^{\leq k}\phi \mid \phi \mathrm{U}\phi$$

where the term "$\top$" means *true*, "ap" is an atomic proposition, $k \in N$, $p \in [0, 1]$, and $\bowtie \in \{<, \leq, >, \geq\}$. The operator "$\wedge$" represents the *conjunction* and "$\neg$" is the *negation* operator, P is the probabilistic operator, and R is the reward operator. Also, "X", "$\mathrm{U}^{\leq k}$", and "U" are the *next*, the *bounded until*, and the *until* temporal logic operators, respectively.

3.3.3 ICPS Recommendation

To provide a secure and valid system model, we designed a recommendation strategy to reinforce the model's system. It is based on the defense-attack tree by extending Markov Decision Attack Trees with counter-measures. Each path of the attack tree is labelled with a countermeasure. Formally a recommendation, say r, is the tuple $\langle \mathcal{M}, \mathscr{C}, \Phi \rangle$, where:

- $\mathcal{M}$ is the attack tree instantiated from the MDP.
- $\mathscr{C}$ is a finite set of countermeasures $c_1, \cdots, c_i, \cdots c_n$
- $\Phi : P \longrightarrow C$ returns for a path p_i a countermeasure c_i. P is the set of paths and sub-path proper to $\mathcal{M}$.

Figure 4 depicts an Markov Decision Attack Tree extended by countermeasures. The red node is the goal of the attack, blue nodes are the sub-goals or steps of an attack whereas the green nodes are the countermeasures for each path of an attack. R is the reward to achieve the state of a node. $(\mathcal{P})$, $(1 - \mathcal{P})$, (1) are the probabilities to execute an action.

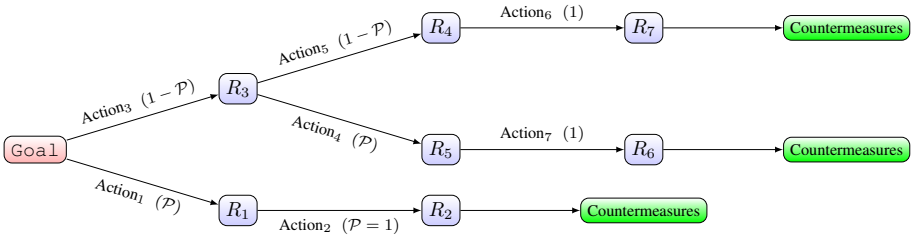

Figure 4: Markov Decision Attack-Defense Tree.

4 Experimental results

This section describes the implementation of our approach and its validation on two real use cases. The experiments were carried out on a PC with Intel Core i5-7200U @ 2.50GHz processor and a RAM of 8.00 GB.

Tool Architecture

Figure 5 depicts the overall architecture of the proposed framework. It implements four packages: (i) the user interface to interact within the kernel of the tool, (ii) the model builder to create and composite the system model and the attacker, (iii) the verification engine to check and reinforce the system in case of successful attacks, and (v) the library that contains templates, attackers, and countermeasures.

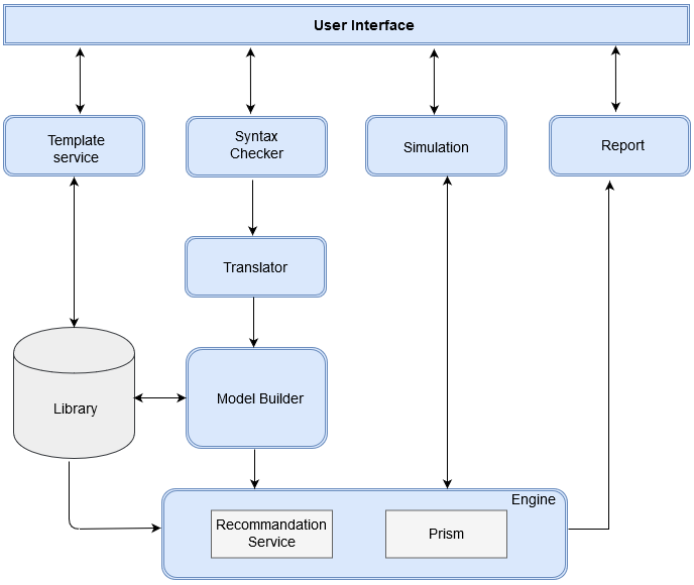

Figure 5: ICPS Tool Architecture.

User interface

The graphical interface allows a user to interact with our tool through four services. *Template service* is connected with the library and it is used to create the system model by relying on specific templates like nodes and protocols. *Syntax Checker* checks the syntax of our language which is used to define the model service. The language is defined in Section 3.1 and we will see how to model a system with it in our case studies. *Simulation* shows the execution of the system and *Report* presents information about the generated PRISM model, the attacks scenarios, attacks impacts, and countermeasures.

Model builder

The main objective of this component is to build the system model in interaction with the attacker. We build the model system from templates and nodes behaviours after that we compose the model system with the attacker through synchronization channels.

Engine

It is the main part in our tool, it contains two components, *Model checker* to verify the security requirements and to generate the attacks scenarios and *Recommendation service* to reinforce the system model by proposing a list of suitable countermeasures.

Library

It contains the templates of nodes, protocols, countermeasures and attack models.

4.1 Case study 1: Utah water-supply

The proposed framework is validated on Utah water-supply[2] system (see Figure 6), which is designed by Logan to enable building an interconnection between Trenton and Amalga systems to guarantee water supply during emergency situations. This system connects numerous sensors to monitor the pressure from water sources to tanks and from tanks to both towns by measuring tank level, chlorine water level, flood conditions and chlorine gas leaks at the building hosting the system. Also, the system controls a set of pumps and valves.

Figure 6: Water-Supply system.

We parameterized the system with a set of variables of type Boolean and integer to define the status of the monitoring and control dashboard. The set of Boolean variables that controls the pumps and valves is $\mathcal{V}_p = \{Pump, Valve_1, Valve_2, SysRun\}$, where $SysRun$ is used to initiate the system, $Pump$ and $Valve_1$ defined to fill the tank, and $Valve_2$ introduced to distribute the water from the tank to the agglomeration. To better monitor the status of sensors, $\mathcal{V}_d$ is used to define each flag and data that evaluate a sensor's measure, where: $\mathcal{V}_d = \{TankLevel, Flood, ChlLeaks, ChlInwater, Pres\}$. $TankLevel$ returns the level of the tank, $Flood$ evaluates if there is any flooding in the building, $ChlLeaks$ to check if there is chlorine gas leaks at the building, $ChlInwater$ measures the level of the chlorine in the water, and Pre returns the measured pressure

[2]https://s.campbellsci.com/documents/us/case-studies/64utah-SCADA.pdf

of the water. Further in an emergency situation, the worker can manually start or stop the pump.

The formal model

In the water-supply system presented in Figure 6, two servers s_1 and s_2 are accessible through the client c (a unique client deploying multi-protocols). $o_p^{s_1}$ is the object of the vector V_p controlled by the OPC-UA server s_1. The $o_d^{s_2}$ is the object of the vector V_d controlled by the Modbus server s_2. The servers s_1 and s_2 control the objects $o_p^{s_1}$ and $o_d^{s_2}$ through the PLC p. Herein, we describe briefly the behaviours of the client C, the servers s_1, s_2, and one physical node ($e.g.$ valve).

The `ICPS` Model

- The behaviour of the client c.
  $Beh(c) = \texttt{Start}.(\texttt{Receive}_{s,s}([\![o_d]\!], s_2).\texttt{Update}_{s,s}(o_p, [\![o_p^{'}]\!]).\texttt{Send}_{s,s}(s_1, o_p^{'})$
  $+\texttt{Receive}_{s,s}([\![o_d]\!], s_2).\texttt{Update}_{s,s}(o_d, [\![o_d^{'}]\!])+\texttt{Send}_{s,s}(s_1, o_p)).\texttt{Terminate}.$

- The behaviours of the servers.
  $Beh(s) = \texttt{Start}.(\texttt{Receive}_{s,s}([\![o]\!], c).\texttt{Send}_{o,s}(o, s)$
  $+ \texttt{Receive}_{o,s}([\![o]\!], o).\texttt{Send}_{s,s}(c, [\![o]\!])).\texttt{Terminate}.$

- The behaviour of the valve.
  $Beh(v) = \texttt{Start}.(\texttt{Receive}_{o,o}([\![o]\!], p).\texttt{Unlock}_o(v, port)$
  $+ \texttt{Receive}_{o,s}([\![o]\!], p).\texttt{Lock}_o(v, port)).\texttt{Terminate}.$

The technical attacker

We consider here an attacker that gains the access to the network with the ability to interact with an employee who has access to the room of control. Herein, we describe briefly the different parts and the ability of this attack.

Knowledge The attacker knowledge consists of the information about the architecture of the network without having any private keys or any physical access. Also he has the algorithm to encrypt, decrypt, and hash messages.

Personality We do not define the personality of the attacker because we will not cover the security analysis related to the social aspect within this attack. We will consider it in the next attack's case.

Skills In this case study we have defined only the *communication attacks* as a skills for the attacker. So, according to his knowledge he can intercept, replay and create messages, encrypt and decrypt if he has a private key. Table 2 specifies the attacker skills. We have used SPi-calculus to formalise the skills of the attacker . $(v\ i)$ allows to add, save, update or get data from his knowledge.

Table 2: Specification of the attacker's skills

$$
\begin{aligned}
A = \quad & (v\ i)(canal(x).!(\bar{i}\langle x\rangle.0 \ +\ i(m).0 \ +\ i(pk).0 \ +\ i(sk).0 \\
& + i(m_1).i(m_2).\bar{i}(\langle m_1, m_2\rangle).0 \\
& + case\ x\ of\ \langle m_1, m_2\rangle\ in\ \bar{i}\langle m_1\rangle.0 \\
& + case\ x\ of\ \langle m_1, m_2\rangle\ in\ \bar{i}\langle m_2\rangle.0 \\
& + i(\langle m_1, m_2\rangle).i(m_3).\bar{i}(\langle m_3, m_2\rangle).0 \\
& + i(\langle m_1, m_2\rangle).i(m_3).\bar{i}(\langle m_1, m_3\rangle).0 \\
& + case\ x\ of\ \{[m]\}_{sk}\ in\ \bar{i}\langle m\rangle.0). \\
& (\overline{canal}\langle m\rangle.0 + \overline{canal}\langle[\{H(m)\}]_{sk}\rangle.0))
\end{aligned}
$$

Goal The goal of the attacker is to dysfunction the system by not filling the tank.

Analysis The attacker objective is described as a Markov decision tree and to calculate the reward for each state we use a weight sum method. Table 3 describes the selected criterion. **D** is the expected probability to not detect the attack step, 1 means no risk to detect whereas 0 means there is a big risk to detect the attack. **F** means the feasibility of the attack, 1 means that the attack is feasible and 0 not. **S** represents if the attacker has the skills to realize this attack step or no, 1 means true and 0 false.

Table 3: Weight sum method settings for the attacker.

Action/criterion	**D**	**F**	**S**
	0.25	0.25	0.5
Intercept	1	1	1
Create msg (Modbus)	0.5	1	1
Create msg (OPC-UA)	0	0	0
Modify msg (Modbus)	0.5	1	1
Modify msg (OPC-UA)	0	0	0
Replay msg (Modbus)	0.5	1	1
Replay msg (OPC-UA)	0	0	0
Block msg	0	1	1
Social engineering	1	0	0

The Socio-technical attacker

Here we consider the attacker as an employee who works in this water-supply. For his knowledge and skills, he has the information about all the system and the access control of SCADA client. Further for his personality, he has a high factor of Neuroticism which means that he is vulnerable and he could be exploited by social engineering technique.

Security requirements

For this case study, we exhibit the following properties:

- The attacker cannot stop the system.
  $\Phi_1 : Pmax =?[F(SysRun = 0 \ \& \ Attack = 1)]$
- The attacker cannot stop the pump.
  $\Phi_2 : Pmax =?[F(Pump = 0 \ \& \ Tank = 1)]$
- The attacker cannot open the valve and the tank is full.
  $\Phi_3 : Pmax =?[F(Valv1 = 1 \ \& \ Tank = 2)]$

Results

The results, obtained from the verification of the code in Appendix 6.3, are summarized in Table 4 where the symbol ✓ means an attack has been found and the symbol ✗ means that the property is safe. Further, we show the verification of each property with respect to the category of the attack. For the technical attack, the obtained MDP model has 215380 states (1 initial), 1169586 transitions with 997282 choices. Whereas the obtained MDP model for the socio-technical attack has 99220 states (1 initial), 514008 transitions with 434632 choices.

Table 4: The verification results.

Action/ Criterion	Technical Attacker	Verification Time (secs)	Social-Technical Attacker	Verification Time (secs)
Φ_1	✗	0.365	✗	0.115
Φ_2	✓	0.226	✗	0.088
Φ_3	✓	0.377	✗	0.092

In theory, the technical attacker cannot violate the property Φ_1 because the OPC-UA server controls the $SysRun$ variable. Thus, the only way to break the property Φ_1 is to manipulate the social attacker, which obviously is not possible as the technical attacker does not have social engineering skills. As well the social attacker does not violate any property because we supposed that he could violate the properties only if he was manipulated by social engineering skills. The result of attacks proper to $\Phi 2$ in Figure 7 shows the convergence of the probability evaluation from 0 to 0.15 after two steps, then it increases up to the maximum value of 0.25 after 10 steps. This result shows that the risk is low when the system is turned on. Compared to the property $\Phi 1$, the probability of attacks for $\Phi 2$ increases linearly to achieve a maximum of 0.225.

Figure 8 depicts the sequence diagram of the attacker violating $\Phi 3$. The Modbus server sends a message to inform a client that the tank is full, and therefore the valve needs to be closed. However, the attacker intercepts the message and modify it in order to leave the valve open while the tank is full. This experiment shows that, even if the non-secure protocols control only the nodes that are not more sensitive than the other nodes, this can create vulnerabilities and destabilize the system.

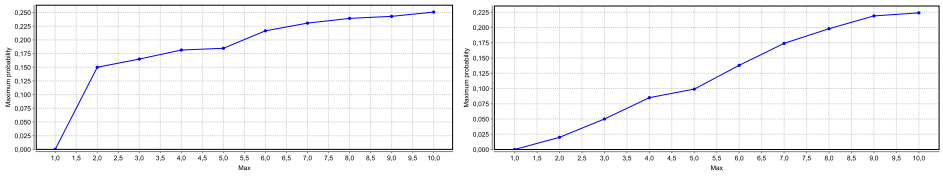

Figure 7: Attack success for Φ_1 (*left*) and Φ_2 (*right*).

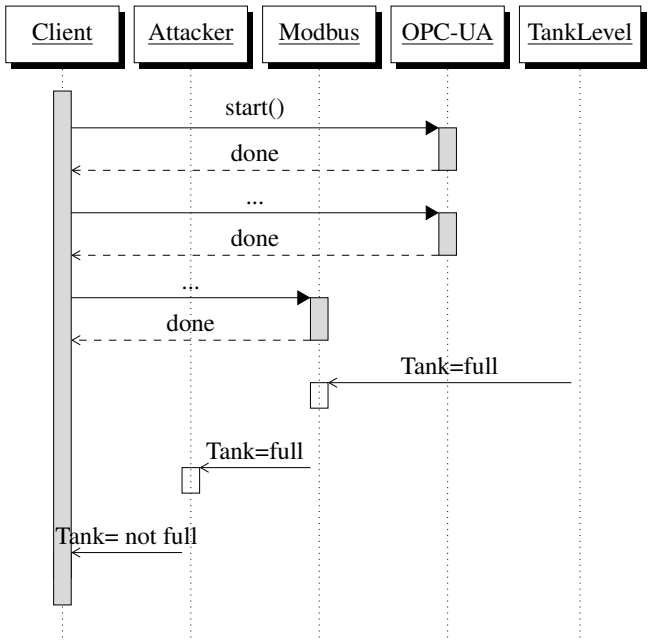

Figure 8: Attack scenario against Φ_3

Countermeasures As shown in Figure 9, the origin of the security breach was originally related to the specifications/implementation of Modbus protocol. Unfortunately, this cannot be changed to OPC-UA protocol as the most deployed equipment (sensors or other) requires the use of Modbus. Consequently, as output, the recommendation system suggests the use of Modbus protocol under the VPN to guarantee better security of the running system.

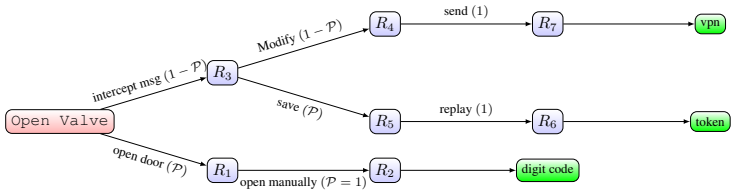

Figure 9: Markov Decision Attack-Defense Tree proper to $\Phi 3$.

4.2 Case study 2: Maroochy Shire Sewage Spill

Maroochy Shire is located about 100 kilometers north of the Queensland State Capital of Brisbane. It has 880 kilometers of gravity sewers treating an average of 35 million liters/day. The Maroochy Water Services Sewerage SCADA system consists of 142 Sewage Pumping Stations, and each pumping station had a computerized system capable of receiving commands from a central control center (master station) and transmitting signals back to the center. The communication between pumping stations and the control center was through a private two-way radio system. Figure 10 shows the system architecture and the connection between its components. The control of the pumping stations can be through the main SCADA station or through one of the pumping stations access points.

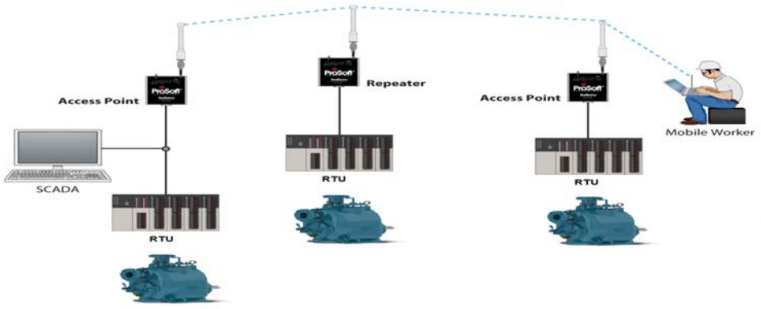

Figure 10: The Maroochy Water Services Sewerage System.

The formal model

In the Maroochy Water Services presented in Figure 10, p_i is a pump, s^i_j a sensor that measures the status of the pump p_i, r_k a remote terminal unit (RTU), m a master station, a an access point, and h an agent who uses the access points with a laptop to manage the system. We used $Beh(m)$ to describe the behaviour of the master station m where it can receive or send a command from/to RTU r_i. $Beh(r_i)$ represents the behaviour of the RTU r_i where it can communicate with the master station m, another RTU r_j, a pump p_j, an access point a_j or a sensor s_j. $Beh(p_i)$ is the behaviour of the pump, it could receive a command o_{r_j} from an RTU to change its state to start or stop

running. $Beh(s_i)$ represents the behaviour of the sensor to receive the status of the pump and send it to the RTU. $Beh(h_i)$ is the behaviour of an agent using the access point to receive and send commands.

- The behaviour of the master station m.
  $Beh(m) = \texttt{Start}.(\texttt{Receive}([\![o_{r_i}]\!], r_i).\texttt{Update}(o_{i_m}, [\![o_{r_i}]\!]) + \texttt{Send}(r_i, [\![o_{i_m}]\!])$
  $+ \texttt{Send}(r_j, [\![o_{j_m}]\!])).\texttt{Terminate}$
- The behaviours of the RTU r_i.
  $Beh(r_i) = \texttt{Start}.(\texttt{Receive}([\![o_{m_i}]\!], m) + \texttt{Receive}([\![o_{r_j}]\!], r_j) + \texttt{Receive}([\![o_{a_j}]\!], a_j)$
  $+ \texttt{Receive}([\![o_{p_j}]\!], p_j).\texttt{Update}(o_i, [\![o_i]\!]).\texttt{Send}(m, [\![o_{r_i}]\!]) + \texttt{Send}(r_j, [\![o_{r_j}]\!])$
  $+ \texttt{Send}(p_k, [\![o_{r_i}]\!])).\texttt{Terminate}.$
- The behaviour of the pump p_i.
  $Beh(p_i) = \texttt{Start}.(\texttt{Receive}([\![o_{r_j}]\!], r_j).(\texttt{Unlock}(p_i, o_{p_i}) + \texttt{Lock}(p_i, o_{p_i}))).\texttt{Terminate}.$
- The behaviour of the sensor S_i.
  $Beh(s_i) = \texttt{Start}.\texttt{Receive}([\![s_{p_j}]\!], p_j).\texttt{Update}(s_i, [\![s_{p_j}]\!]).\texttt{Send}(r_i, [\![s_{p_i}]\!]).\texttt{Terminate}.$
- The behaviour of the agent h_i.
  $Beh(h_i) = \texttt{Start}.(\texttt{Receive}([\![o_{a_j}]\!], a_j).\texttt{Update}(o_i, [\![o_{a_j}]\!]).(\texttt{Send}(a_{j'}, [\![o_i]\!]$
  $+ \texttt{Receive}([\![o_{a_j}]\!], a_j)).\texttt{Terminate}.$

The insider technical attacker

In this scenario, we model the attacker who hacked the system in the real world.

Knowledge The real attacker was an old employee with a strong knowledge about the system functionalities and the used communication protocols.

Personality The attacker has a high factor of Neuroticism.

Skills The attacker has already installed the software in his laptop as well as he has the knowledge to connect to the system, to change the state of all the pumps, and either to enable/disable alarms. Table 5 specifies the insider attacker skills.

Table 5: Specification of the insider technical attacker.

$$
\begin{aligned}
A = \quad & (v\ i)(canal(x).!(\bar{i}\langle x\rangle.0 \ +\ i(m).0 \\
& + i(m_1).i(m_2).\bar{i}(\langle m_1, m_2\rangle).0 \\
& + case\ x\ of\ \langle m_1, m_2\rangle\ in\ \bar{i}\langle m_1\rangle.0 \\
& + case\ x\ of\ \langle m_1, m_2\rangle\ in\ \bar{i}\langle m_2\rangle.0 \\
& + i(\langle m_1, m_2\rangle).i(m_3).\bar{i}(\langle m_3, m_2\rangle).0 \\
& + i(\langle m_1, m_2\rangle).i(m_3).\bar{i}(\langle m_1, m_3\rangle).0). \\
& (\overline{canal}\langle m\rangle.0)
\end{aligned}
$$

Goal. The main objective of the attacker is to take the control over the station pumping.

Analyzer. Table 6 describes the expected probability to not detect the attack step **D**, the feasibility of the attack **F**, and the skills **S**.

Table 6: Weight sum method settings for the insider technical attacker.

Action/criterion	**D**	**F**	**S**
	0	1	1
Create msg	0	1	1
Replay msg	0	1	1
DoS attack	0.5	1	0

The external technical attacker #1

This attacker has less knowledge and skills than the insider attacker.

Knowledge This attacker has a knowledge about the fourth pumping station as well as the radio communications.

Personality In this scenario of attack, we exclude the importance of the attacker personality.

Skills This attacker knows how to find the radio frequency and how to receive and send data over the found channel. He can do only DoS attack to saturate the channel. Table 7 specifies the skills of the external technical attacker #1.

Table 7: Specification of the external technical attacker #1.

$$A = \quad (v\ i)((canal(x).\bar{i}\langle x\rangle.0\ ) + (\ i(m).\ !(\overline{canal}\langle m\rangle.0)))$$

Analyzer. Table 8 describes the selected criterion **D**, **F**, and **S** previously defined.

Table 8: Weight sum method settings for the external technical attacker #1.

Action/criterion	**D**	**F**	**S**
	0.5	1	1
Read msg	1	1	1
DoS attack	0.5	1	1

The external technical attacker #2

We model this attacker with another kind of knowledge and skills than the first one.

Knowledge We consider that this attacker has no knowledge about the pumping stations.

Personality We exclude in this scenario the personality description and impact on the attack goal.

Skills As described in Table 9, the attacker has good skills on malware development, reverse engineering, network analysis and social engineering.

Table 9: Specification of the external technical attacker #2.

$$
\begin{aligned}
A = \quad & (v\ i)((canal_{chan}(x) + canal_{hum}(x)).!(\bar{i}\langle x\rangle.0\ +\ i(m).0 \\
& + \mathrm{i}(\mathrm{m}_1).i(m_2).\bar{i}(\langle m_1, m_2\rangle).0 \\
& + case\ x\ of\ \langle m_1, m_2\rangle\ in\ \bar{i}\langle m_1\rangle.0 \\
& + case\ x\ of\ \langle m_1, m_2\rangle\ in\ \bar{i}\langle m_2\rangle.0 \\
& + i(\langle m_1, m_2\rangle).i(m_3).\bar{i}(\langle m_3, m_2\rangle).0 \\
& + i(\langle m_1, m_2\rangle).i(m_3).\bar{i}(\langle m_1, m_3\rangle).0). \\
& ((\overline{canal_{chan}}\langle m\rangle + \overline{canal_{hum}}\langle m\rangle).0))
\end{aligned}
$$

Analyzer. Compared to the previous cases, we consider **F** as a probabilistic value (Table 10) since the feasibility of a social engineering attack does not depend only on the skills of the attacker but also on the reactions of victims. The feasibility for the social engineering attack is a variable x since he does not know the personalities of the employees. In this case, we consider two scenarios depending the values of x that can be either 0 (scenario 1) or 0.5 (scenario 2). Also, we consider that we have five employees and one of them has a high factor of Neuroticism which means he is vulnerable and he could be exploited by social engineering techniques. Each one of the employee has the information about all the system and the access control of SCADA client. Also they have a laptop and the software to access to the SCADA system through the access points.

Table 10: Weight sum method settings for the external technical attacker #2.

Action/criterion	**D**	**F**	**S**
	1	0.5	1
Modify Msg	1	1	1
Social engineering attack	1	x	1

Security requirements

For this case study, we have the following properties:

- Pump 4 was running when it should not.
  $\Phi_1 : Pmax =?[F(Pump_4 = 1 \ \& \ Sensor_4 = 0)]$
- Pump 4 was not running when it should not.
  $\Phi_2 : Pmax =?[F(Pump_4 = 0 \ \& \ Sensor_4 = 1)]$
- Could the attacker stop another pump rather than pump 4.
  $\Phi_3 : Pmax =?[F(Pump_{i\neq 4} = 0 \ \& \ Sensor_{i\neq 4} = 1)]$

Results

Table 12 and Table 13 show the verification results of the three properties for the described attacks, respectively. Table 11 refers to the size of the constructed MDP proper to each attack in terms of states, transitions, and choices. From the obtained results, we found that the insider attacker and the social-Technical attacker violate all the properties since both of them have all the knowledge, skills and the software to control the system. Further, the external technical attacker #1 could not violate the third property because he has only DoS attack as skills. Thus, he could not disturb the other pump stations. Furthermore, the external technical attacker #2 could not violate all the properties because he do not have the skills and the knowledge about the system to violate the properties but he succeeded when mastering social-technical attack techniques.

Table 11: The model complexity in the presence of the different attackers.

Action/ Criterion	Insider Attacker	External-Technical Attacker #1	External-Technical Attacker #2 (scenario #1)	External-Technical Attacker #2 (scenario #2)
States	29520	45980	59040	117260
Transitions	259200	440924	518400	1029600
Choices	141120	220220	282240	560560

Table 12: The verification results in the presence of the insider and the external technical #1 attackers.

Action/ Criterion	Insider Attacker	Verification Time (secs)	External-Technical Attacker #1	Verification Time (secs)
Φ_1	✓	0.109	✓	0.393
Φ_2	✓	0.221	✓	0.491
Φ_3	✓	0.301	✗	0.537

Figure 11 shows the success of the insider attacker and the external-technical attacker #1 for Φ_1. The probability evaluation of the insider attack converges to 0.138

Table 13: The verification results in the presence of the external technical attacker #2.

Action/ Criterion	External Technical Attacker #2 (scenario #1)	Verification Time (secs)	External Technical Attacker #2 (scenario #2)	Verification Time (secs)
Φ_1	✗	0.222	✓	0.051
Φ_2	✗	0.297	✓	0.077
Φ_3	✗	0.311	✓	0.092

and to 0.327 for the external-technical attacker #1. Further, Figure 12 shows the success of the social technical attacker for Φ_2 with a maximum probability value of 0.362 and 0.404 for Φ_3 (*right*).

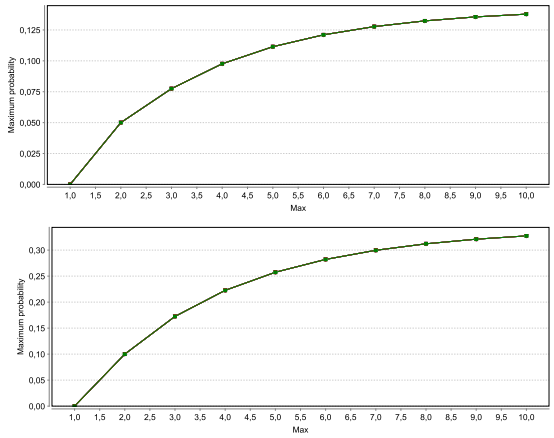

Figure 11: The success of insider Attacker (*left*) and the External-technical Attacker #1 for Φ_1 (*right*).

Figure 13 depicts the sequence diagram of violating $\Phi 3$ by the external technical attacker #2 (scenario #2). The attack start by sending emails (social engineering attack) to all agents. This phishing email contains a link to download a fake update of the system control software. When the agent click on the link, a malware will be download instead of the trusted software update. Then, the malware starts recording everything (packets, keyboard inputs, etc) and send it back to the attacker. After analyzing the received data, he patches the malware to update the commands. After that, the malware intercepts the command, modifies and sends it to the RTU. Consequently, the pump will be stopped.

Countermeasures Figure 14 shows the countermeasures generated for the violation of $\Phi 2$ by the insider attacker (the bottom path), the external attacker #1 (the middle path), and the external attacker in scenario #2 (the upper path). Each path is an attempt from all possible flows of each attack. The origin of the security breaches were initially

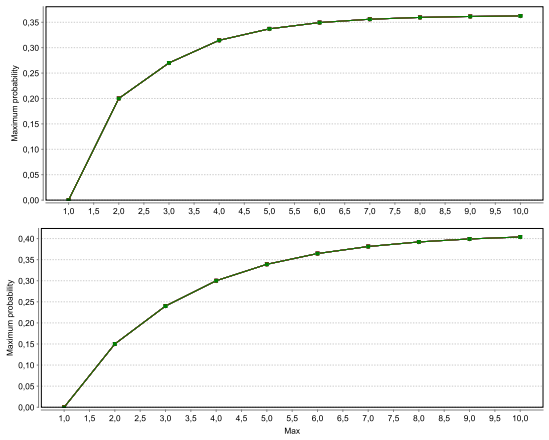

Figure 12: The success of the social technical attacker for Φ_2 (*left*) and Φ_3 (*right*).

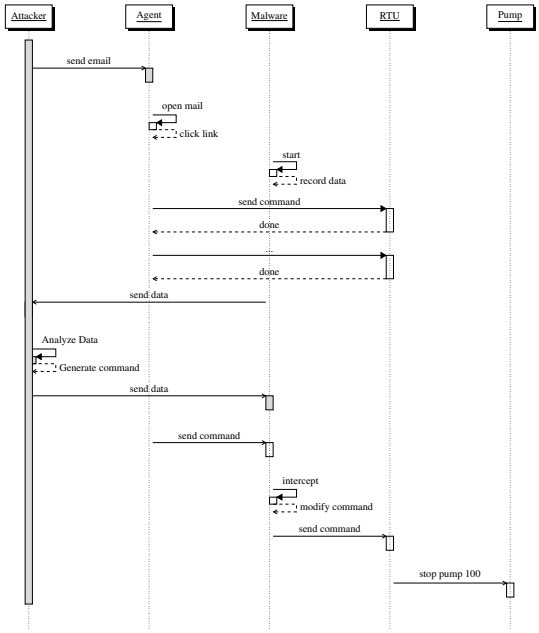

Figure 13: Attack scenario against Φ_3.

related to two points: 1) the insider attacker has too much power, and 2) the use of the radio communication. The generated countermeasure rely on a defense-in-depth strategy. First, it proposes the use of access control to alleviate an attack. In addition, it suggests an identification and authentication mechanism to harden the access control and issuing radio commands. Further, it prefers the use of the system monitoring to

ensure the real time trustworthy and functionality of the system. Also, it recommends a system and information integrity to avoid sending false commands and controlling pumps. Furthermore, it proposes to periodically perform personality tests to assess the employee behaviours and plan training to overcome social engineering attacks.

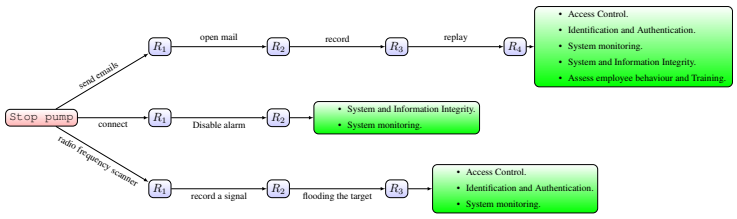

Figure 14: Markov Decision Attack-Defense Tree for $\Phi 2$.

5 Conclusion

This paper sets the fundamentals needed for a fully automatic framework to correctly model and analyze safely security in ICPS. We provided details of the formalism needed for a complete ICPS architecture that captures the main structures and possible behaviour of ICPS entities by covering the physical and information infrastructures, services, assets, social actors, and also their activities and interactions. Further, the formalism included the intruder with different powers regarding the aspect of each ICPS entity. The execution of an action has a cost and guided by probabilities under contextual conditions. At this end, the reinforcement learning aims to help an attack to select an optimal sequence of actions in terms of reducing the cost and maximizing the reward. The proposed formalism has also a rich and flexible semantics, which is used to capture the ICPS functional and security requirements expressing the possibility, the likelihood, and the cost of actions. This is designed to be easy for other extensions/refinements and to automatically carry the functional correctness and security analysis. This is done using the proposed algorithm that transforms an ICPS model into the input language of PRISM so to be checked against the requirements expressed in PCTL within the presence of attacks. Finally, the effectiveness of the framework is validated on two case studies that shows the impact of attacks and how they are mitigated.

The proposed work sets the stage for further improvements. First, we intend to refine the contextual conditions of the proposed formalism, generate automatically the security requirements needed for verification, and provide a countermeasure mechanism that automatically reinforces the ICPS security. Also from a theoretical point of view, we needed to automate the presented proofs for each developed step in a proof assistant (e.g. Coq). Further, we need to accelerate the analysis procedure by introducing the parallelism for probabilistic verification. Furthermore, we intend to implement the proposed framework as a full standing tool and later validate it on different case studies.

References

[1] S. Adepu and A. Mathur. Generalized attacker and attack models for cyber physical systems. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 283–292, June 2016.

[2] Anand Agrawal, Chuadhry Mujeeb Ahmed, and Ee-Chien Chang. Poster: Physics-based attack detection for an insider threat model in a cyber-physical system. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, pages 821–823. ACM, 2018.

[3] Rajeev Alur. *Principles of Cyber-Physical Systems*. The MIT Press, 2015.

[4] Yosef Ashibani and Qusay H. Mahmoud. Cyber physical systems security: Analysis, challenges and solutions. *Computers & Security*, 68:81–97, 2017.

[5] G. Bakirtzis, B. T. Carter, C. R. Elks, and C. H. Fleming. A model-based approach to security analysis for cyber-physical systems. In *2018 Annual IEEE International Systems Conference (SysCon)*, pages 1–8, April 2018.

[6] Carmen Cheh, Ahmed M. Fawaz, Mohammad A. Noureddine, Bin Bin Chen, William G. Temple, and William H. Sanders. Determining tolerable attack surfaces that preserves safety of cyber-physical systems. *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 125–134, 2018.

[7] Y. Chen, S. Kar, and J. M. F. Moura. Dynamic attack detection in cyber-physical systems with side initial state information. *IEEE Transactions on Automatic Control*, 62(9):4618–4624, 2017.

[8] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[9] Gabriele Lenzini, Sjouke Mauw, and Samir Ouchani. Security analysis of socio-technical physical systems. *Computers & Electrical Engineering*, 47:258–274, 2015.

[10] Xiaoxue Liu, Jiexin Zhang, and Peidong Zhu. Modeling cyber-physical attacks based on probabilistic colored petri nets and mixed-strategy game theory. *International Journal of Critical Infrastructure Protection*, 16:13 – 25, 2017.

[11] R. Mitchell and I. Chen. Modeling and analysis of attacks and counter defense mechanisms for cyber physical systems. *IEEE Transactions on Reliability*, 65(1):350–358, 2016.

[12] Samir Ouchani. Ensuring the Functional Correctness of IoT through Formal Modeling and Verification. In *Model and Data Engineering - 8th International Conference, MEDI 2018,LNCS, Springer, Proceedings*, pages 401–417, 2018.

[13] Samir Ouchani and Gabriele Lenzini. Generating attacks in sysml activity diagrams by detecting attack surfaces. *Journal of Ambient Intelligence and Humanized Computing*, 6(3):361–373, Jun 2015.

[14] Maxime Puys, Marie-Laure Potet, and Abdelaziz Khaled. Generation of applicative attacks scenarios against industrial systems. In Abdessamad Imine, José M. Fernandez, Jean-Yves Marion, Luigi Logrippo, and Joaquin Garcia-Alfaro, editors, *Foundations and Practice of Security*, pages 127–143, Cham, 2018. Springer International Publishing.

[15] Marco Rocchetto and Nils Ole Tippenhauer. Cpdy: Extending the dolev-yao attacker with physical-layer interactions. In Kazuhiro Ogata, Mark Lawford, and Shaoying Liu, editors, *Formal Methods and Software Engineering*, pages 175–192, Cham, 2016. Springer International Publishing.

[16] Marco Rocchetto and Nils Ole Tippenhauer. On attacker models and profiles for cyber-physical systems. In Ioannis Askoxylakis, Sotiris Ioannidis, Sokratis Katsikas, and Catherine Meadows, editors, *Computer Security – ESORICS 2016*, pages 427–449, Cham, 2016. Springer International Publishing.

[17] Giedre Sabaliauskaite and Aditya P. Mathur. Aligning cyber-physical system safety and security. In Michel-Alexandre Cardin, Daniel Krob, Pao Chuen Lui, Yang How Tan, and Kristin Wood, editors, *Complex Systems Design & Management Asia*, pages 41–53, Cham, 2015. Springer International Publishing.

[18] Gaddadevara Matt Siddesh, Ganesh Chandra Deka, Krishnarajanagar GopalaIyengar Srinivasa, and Lalit Mohan Patnaik. *Cyber-Physical Systems: A Computational Perspective*. Chapman & Hall/CRC, 2015.

6 Appendices

6.1 Operational semantics

For anode n, rules LCK, UCK, and MOV describe respectively locking, unlocking, and moving a node. TRT is to terminate the process.

$$\frac{B_n = lock(o_1, o_2).B_n^{'} \wedge \neg locked(o_1) \wedge o_1, o_2 \in cont(n) \wedge p \in attr(o_1) \cap attr(o_2)}{\langle id_n, -, < -, \{o_1, \neg locked(o_1)\} >, -\rangle \xrightarrow{lock(o_1,o_2)} \langle id_n, B_n^{'}, < -, \{o_1, locked(o_1)\} >, -\rangle} \text{ LCK}$$

$$\frac{B_n = lock(o_1, o_2).B_n^{'} \wedge locked(o_1) \wedge o_1, o_2 \in cont(n) \wedge p \in attr(o_1) \cap attr(o_2)}{\langle id_n, -, < -, \{o_1, locked(o_1)\} >, -\rangle \xrightarrow{lock(o_1,o_2)} \langle id_n, B_n^{'}, < -, \{o_1, \neg locked(o_1)\} >, -\rangle} \text{ UCK}$$

$$\frac{B_n = \texttt{Move}(l, l').B_n^{'} \wedge loc(n) = l \wedge locked(lock(l))}{\langle id_n, -, < loc(n) = l, -, -, -\rangle \xrightarrow{up_x} \langle id_n, B_n^{'}, < loc(n) = l', -, -, -\rangle} \text{ MOV}$$

For the composition operators, we define the following semantic rules where P1 and P2 represent the parallel composition rules, COM is the communication one, N1 and N2 are the non-deterministic rules. Further, G1 and G2 represent the guarded choice derivation rules. Finally PB1 and PB2 are the derivation rules for the probabilistic choices. To simplify the derivation rules, we consider n_1 and n_2 two different nodes.

$$\frac{n_1 \xrightarrow{\alpha_1} n_1'}{n_1||n_2 \xrightarrow{\alpha_1} n_1'||n_2}\ \text{P1} \qquad \frac{n_2 \xrightarrow{\alpha_2} n_2'}{n_1||n_2 \xrightarrow{\alpha_1} n_1||n_2'}\ \text{P2} \qquad \frac{n_1 \xrightarrow{\alpha_1} n_1'}{n_1 + n_2 \xrightarrow{\alpha_1} n_1' + n_2}\ \text{N1}$$

$$\frac{n_2 \xrightarrow{\alpha_2} n_2'}{n_1 + n_2 \xrightarrow{\alpha_1} n_1 + n_2'}\ \text{N2} \qquad \frac{n_1 \xrightarrow{\alpha_1,g} n_1'}{n_1 +_g n_2 \xrightarrow{\alpha_1,g} n_1'}\ \text{G1} \qquad \frac{n_2 \xrightarrow{\alpha_2,g} n_2'}{n_1 +_{\neg g} n_2 \xrightarrow{\alpha_2} n_2'}\ \text{G2}$$

$$\frac{n_1 \xrightarrow{\alpha_1,p} n_1'}{n_1 +_p n_2 \xrightarrow{\alpha_1,p} n_1'}\ \text{PC1} \qquad \frac{n_2 \xrightarrow{\alpha_2,p} n_2'}{n_1 +_{1-p} n_2 \xrightarrow{\alpha_2,p} n_2'}\ \text{PC2} \qquad \frac{n_1 \xrightarrow{\alpha_1} n_1' \quad n_2 \xrightarrow{\alpha_2} n_2'}{n_1|n_2 \xrightarrow{\mathcal{C}(\alpha_1,\alpha_2)} n_1'|n_2'}\ \text{COM}$$

6.2 Transformation Function

$$\mathcal{T}_P(\alpha) = \begin{cases} [SRT_o]\neg o_s \rightarrow (o'_s = \top); \\ \texttt{iff:Start} \in \Sigma_o \\ \\ [TRT_o]\neg o_t \rightarrow (o'_t = \top); \\ \texttt{iff:Terminate} \in \Sigma_o \\ \\ [Syn_{o_2}]o_{o_2} \wedge o_{1_{o_3}} \wedge \neg p_{o_2} \wedge \neg p_{o_3} \rightarrow (o'_2 = o_2); \\ [Syn_{o_2}]o_{o_2} \wedge o_{1_{o_3}} \wedge \neg p_{o_2} \wedge \neg p_{o_3} \rightarrow (o'_3 = o_2); \\ \texttt{iff:Send}(o_1, o_2) \in \Sigma_o^{o_1}, \texttt{Receive}(o_3, o_2) \in \Sigma_o^{o_2}. \\ \\ [Tak_{o_1}]l_a = l_o \wedge o_{o_2} \wedge \neg lock_o \wedge p_{o_2} \rightarrow (a'_{o_2} = \top); \\ [Tak_{o_1}]l_a = l_o \wedge o_{o_2} \wedge \neg lock_o \wedge p_{o_2} \rightarrow (o'_{o_2} = \bot); \\ \texttt{iff:Receive}(o, o_2) \in \Sigma^a. \\ \\ [Put_{o_1}]l_a = l_o \wedge a_{o_i} \wedge \neg lock_o \wedge p_{o_i} \rightarrow (a'_{o_i} = \bot); \\ [Put_{o_1}]l_a = l_o \wedge a_{o_i} \wedge \neg lock_o \wedge p_{o_i} \rightarrow (o'_{o_i} = \bot); \\ \texttt{iff:Send}(o, o_2) \in \Sigma^a. \\ \\ [loc_{o_1}]o_{o_1} \wedge o_{o_2} \wedge \neg k_{o_1} \wedge p_{o_1} = p_{o_2} \rightarrow (k'_{o_1} = \top); \\ \texttt{iff:Lock}(o_1, o_2) \in \Sigma_o^o. \\ \\ [loc_{o_1}]o_{o_1} \wedge o_{o_2} \wedge k_{o_1} \wedge p_{o_1} = p_{o_2} \rightarrow (k'_{o_1} = \bot); \\ \texttt{iff:Unlock}(o_1, o_2) \in \Sigma_o^o. \\ \\ [loc_{o_1}]o_{o_1} \wedge o_{o_2} \wedge k_{o_1} \wedge p_{o_1} = p_{o_2} \rightarrow ([\![o_1]\!]' = [\![o_2]\!]); \\ \texttt{iff:Update}(o_1, o_2) \in \Sigma_o^o. \end{cases}$$

For $\alpha \in \Sigma$, we have:

1. $\mathcal{T}_P(\alpha) = \langle l_\alpha, g_\alpha, u_\alpha \rangle$ where l_α is the label, g_α is the guard, and l_α is the update for the command in PRISM $[l_\alpha]g_\alpha \rightarrow (u_\alpha);$.
2. $\mathcal{T}_P(\alpha.\alpha') = \mathcal{T}_P(\alpha) \cup \mathcal{T}_P(\alpha')$.
3. $\mathcal{T}_P(\alpha + \alpha') = \mathcal{T}_P(\alpha) \cup \mathcal{T}_P(\alpha')$ s.t. $g_{\alpha'} = g_{\alpha'} \wedge u_\alpha$.
4. $\mathcal{T}_P(\alpha +_g \alpha') = \mathcal{T}_P(\alpha) \cup \mathcal{T}_P(\alpha')$ s.t. $g_\alpha = g_\alpha \wedge g$ and $g_{\alpha'} = g_{\alpha'} \wedge \neg g$.
5. $\mathcal{T}_P(\alpha +_p \alpha') = \langle l_p, g_\alpha \vee g_\alpha, \langle (p, u_\alpha), (1-p, u_\alpha) \rangle \rangle$.
6. $\mathcal{T}_P(\alpha || \alpha') = \mathcal{T}_P(\alpha) \cup \mathcal{T}_P(\alpha')$.
7. $\mathcal{T}_P(\alpha | \alpha') = \mathcal{T}_P(\alpha) \cup \mathcal{T}_P(\alpha')$ s.t. $l_\alpha = l_{\alpha'}$.

6.3 The Generated PRISM Model

For the security and performance assessment of the water-supply, the tool generates the PRISM code presented in Listing 6.3 that is proper to the model under test. It shows the code fragments of *client*, *modbus*, *opcua*, and *attacker*.

```
// Water-supply system
// Client: Multi protocols
// Servers; Modbus, OPC-UA
// PLC
// Physical nodes
mdp

// Network variables
global Action      : [0..10] init 0;
global Function    : [0..1] init 0;
global Value       : [0..2] init 0;

module Client
       sclient:[0..7] init 0;

       [Sreq]     (sclient=0) -> (sclient'=1);
       [Rreq]     (sclient=1) -> (sclient'=2);
       [SOPCUA]   (sclient=2) -> (sclient'=3);
       [ROPCUA]   (sclient=3) -> (sclient'=4);
       []         (sclient=4) -> (sclient'=5);

       []         (sclient=5) -> (sclient'=6) & (Action'=4) & (Function'=0);
       []         (sclient=5) -> (sclient'=6) & (Action'=5) & (Function'=0);
       [SModbus] (sclient=6) -> (sclient'=7);
       [RModbus] (sclient=7) -> (sclient'=5);
endmodule

module Modbus
       smodbus:[0..3] init 0;

       [Smodbus]    (smodbus=0) -> (smodbus'=1);
       [RPLC]       (smodbus=1) -> (smodbus'=2);
       [SPLC]       (smodbus=0) -> (smodbus'=3);
       [Rmodbus]    (smodbus=2) -> (smodbus'=0);
       [Rmodbus]    (smodbus=3) -> (smodbus'=0);
endmodule

module OPCUA
       sopcua:[0..3] init 0;

       [SOPCUA]   (sopcua=0) -> (sopcua'=1);
       [RPLC]     (sopcua=1) -> (sopcua'=2);
       [SPLC]     (sopcua=2) -> (sopcua'=3);
       [ROPCUA]   (sopcua=3) -> (sopcua'=0);
endmodule

rewards
sAttacker=1 : 1;sAttacker=4 : 0.875;
sAttacker=2 : 0.5;sAttacker=9 : 0.875;
sAttacker=7 : 0.5;sAttacker=14 : 0.875;
sAttacker=12 : 0.5;sAttacker=16 : 0.75;
sAttacker=18 : 0.25;sAttacker=5:  0;
endrewards

module Attacker
       sAttacker:[0..10] init 0;
       Attack: [0..1] init 0;

       //Intercept
       [recieve_modbus] sAttacker=0 -> (sAttacker'=5);
       [send_opc]       sAttacker=0 -> (sAttacker'=5);
       []sAttacker=5 -> 0.097:(sAttacker'=1) + 0.170:(sAttacker'=3) + 0.097:(sAttacker'=6)
        + 0.170:(sAttacker'=8) + 0.097:(sAttacker'=11) + 0.170:(sAttacker'=13)
        + 0.151:  (sAttacker'=15) + 0.048:(sAttacker'=17);
       //Modify Modbus message
       []sAttacker=3 -> (sAttacker'=4) & (Action'=6) & (Function'=0) & (Value'=0) & (Attack'=1);
       []sAttacker=3 -> (sAttacker'=4) & (Action'=6) & (Function'=0) & (Value'=1) & (Attack'=1);
       []sAttacker=3 -> (sAttacker'=4) & (Action'=6) & (Function'=1) & (Value'=0) & (Attack'=1);
       []sAttacker=3 -> (sAttacker'=4) & (Action'=6) & (Function'=1) & (Value'=1) & (Attack'=1);
       [] sAttacker=4 -> (sAttacker'=10);
       [recieve_modbus_1] sAttacker=10 -> (sAttacker'=0);
endmodule
```