Robust stealthy attacks based on uncertain costs and labeled finite automata with inputs

Rabah Ammour\(^1\) Member, IEEE, Said Amari\(^2\), Leonardo Brenner\(^1\), Isabel Demongodin\(^1\) and, Dimitri Lefebvre\(^3\), Senior Member, IEEE

Abstract—This paper deals with the vulnerability analysis of cyber-physical systems subject to malicious actions. For this purpose, the considered system is abstracted as a discrete event system. Labeled finite automata with inputs are used to model the system’s behavior along with the information that circulates in both the input and output channels. In particular, we study here stealthy, i.e., undetectable, cyber-attacks that aim to drive the system from a given normal state to a set of forbidden states. We assume that the attacker has limited resources, i.e., a credit, to insert and delete control and sensors events. The proposed analysis evaluates the costs of such attacks on the controlled system depending on its structure, the cost of the malicious actions and possible uncertainties that may affect those costs. It provides systematic methods that aim to compute attacks of minimal cost and robust attacks that are weakly impacted by uncertainties. A case study representing a manufacturing plant is considered to illustrate the results.

Index Terms—Discrete Event Systems, Automata, Cyber-Physical Systems, Robust Attacks.

I. INTRODUCTION

Cyber-Physical Systems (CPSs) have been widely used in numerous applications such as networked control systems, smart power grids, healthcare systems, advanced communication processes and autonomous transportation networks. CPSs integrate computational and communication capabilities to control and monitor physical processes. The communication and data exchange networks between controllers /supervisors and the operative part of the process increase the vulnerability of CPSs to various types of attacks that may lead to critical and dangerous situations. Examples of cyber attacks include the StuxNet strike on industrial control systems [1], and the spoofing of global positioning systems to capture unmanned aircrafts [2]. As a consequence, vulnerability analysis of CPSs is an increasingly important problem and several works addressed this issue in recent years [3]. In the context of Discrete Event Systems (DESs), automata formalism is suitable to model CPSs to analyze their vulnerability.

Authors of [4] use this formalism and investigate the problem of synthesizing an attack strategy for a given controlled DES by adopting an attacker’s viewpoint. They assume that the attacker is able to manipulate the sensor measurements in order to mislead the controller and drive the system to unsafe or undesirable states without being detected. A similar problem has been addressed by authors of [5] considering actuator attacks where the intruder partially observes the execution of the closed-loop system and can modify the control generated by the supervisor. In the work of [6], attacks on both sensors and actuators layers in networked supervisory control systems are addressed. Thus, the intruder can hide, insert or replace events with the objective to drive the system to reach unsafe states.

Attacks on both sensors and actuators are also considered in [7] but in the discrete-time distributed multi-agent systems framework. The authors show how an attack on a compromised agent can propagate and affect other agents that are reachable from it. An adaptive attack compensator is designed to limit the attack effect and its propagation. Attacks and defense graphs [8], [9] and kill chains [10] are widely used tools to deal with the problem of security assessment due to their great ability to detail network attacks. Basically, the nodes of such graph or chains represent vulnerabilities or devices, and the edges represent the possible evolution within nodes, e.g., the attacker gains more and more privilege in the network by exploiting successive vulnerabilities. Different analysis methods (such as path search, Bayesian networks) are then applied on the graph to assess the vulnerabilities of the network. In the meantime, some approaches have been extended to the stochastic context [11], [12], [13] allowing to quantify, in a probabilistic sense, the attack strategies.

An other concept has been proposed in [14] where a fixed cost has been assigned to each possible attack action. This cost captures the expense (in terms of time, complexity or data packet size) of the attacker when trying to alter the exchanged data in the CPS. This concept makes it possible to model the attacker’s cost constraints and to characterize feasible attacks scenario with respect to intruder’s available resources.

In this paper, we assume that the network security barriers have failed and we consider malicious stealthy actions on both sensors and control events allowing the attacker to drive the system from its current state to a set of target (forbidden or dangerous) states. We refer to such attacks as stealthy moving attacks in the rest of the paper. The context is similar to
In the sense that the attacker is assumed to have a certain credit to manipulate the control symbols sent to the actuators and the output labels returned by the sensors. It is supposed that each action, which can be an insertion or deletion of symbols/labels, has a certain cost. Consequently, stealthy cyber-attacks of limited cost could be considered. The stealthiness characterizes the ability of the attacker to hide its traces and to remain undetectable while it moves the system’s current state. Assigning constant values to the attack costs has been developed in [15] with a vulnerability analysis based on Dijkstra algorithm. In the present work, we introduce uncertain costs represented by intervals that may be used when costs are varying or not perfectly known. This new setting is consistent with many practical situations for which the cost values cannot be exactly estimated. By enlarging the cost intervals, the approach can be implicitly extended to situations that include some unknown costs. This setting needs the use of a new approach based on min-max regret [16].

With this new concept, the main objective of this paper is to evaluate and discuss the vulnerability of the CPS by defining and computing attacks of maximal robustness. Such attacks are characterized by the lowest impact of the uncertainties on their global cost. Note that exploring robust attacks that aim to minimize the maximal regret is interesting not only for the attacker but also for the defender in the sense that this new notion helps to design and refine a defence strategy.

The rest of the paper is organized as follows. Section II is about the motivations and backgrounds. Section III introduces uncertain cost graphs based on labeled finite automata with inputs. Section IV is devoted to the vulnerability analysis of CPS affected by stealthy moving attacks. Section V is a case study and Section VI concludes the paper.

II. BACKGROUNDS

We consider attacks that aim to drive the system from a given (normal) state to a target (forbidden or dangerous) state. Such a moving attack is able to change the information that circulates in both the input and output channels of the system as represented in Figure 1. Consequently, it can replace the true control sequence by a wrong control sequence. In the same time, the attacker is able to erase the traces generated by its malicious actions or to insert wrong traces that are similar to the expected traces to be observed by the user which makes such attacks stealthy. The following assumptions are considered:

- the attacker knows the model of the system,
- the attacker knows (or is able to estimate) the current state of the system to perform the attack,
- the attacker can manipulate (insert or delete) the input symbols and output labels, each action corresponds to a given cost and the attacker has a limited credit to perform its attack.

For the sake of brevity, we restrict the proposed analysis to the insertion of input events and deletion of output labels and consider the cases where the attacker can manipulate all events and labels without any restriction, as far as it has enough credit. It is worth noting that the previous assumptions can be relaxed, in particular, by increasing some cost values.

![Fig. 1: Cyber-physical system under attack](image1)

In our previous work [17], a particular class of Petri nets, called output synchronized Petri nets, has been defined to model the controlled plant and the information that circulates through the CPS. To represent and analyze the behavior of such a model, labeled finite state automaton with inputs is derived from this formalism. It represents the states space of the system as well as the input and output information.

**Definition 1:** A labeled finite automaton with inputs (LFAI) is a 6-tuple \( G = (X, \varepsilon, \delta, x_0, Q, \text{Obs}) \), where

- \( X \) is a finite set of states,
- \( \varepsilon \) is a finite set of symbols (i.e., external input events) and \( E_\lambda = \varepsilon \cup \{ \lambda \} \) where \( \lambda \) is an internal and “always occurring” event,
- \( \delta : X \times E_\lambda \rightarrow X \) is a (possibly partially defined) transition function,
- \( x_0 \in X \) is an initial state,
- \( Q \) is a finite set of labels (i.e., output events) and \( Q_\varepsilon = Q \cup \{ \varepsilon \} \), where \( \varepsilon \) denotes the absence of label,
- \( \text{Obs} : X \times E_\lambda \rightarrow 2^Q \cup \{ \varepsilon \} \) is a labeling function.

We consider that the LFAI is deterministic with respect to the symbols i.e., \( \forall x \in X, \forall e \in E_\lambda, |\delta(x, e)| \in \{0, 1\} \) where \(|.|\) stands for the cardinality of a set. If \( |\delta(x, e)| = 1 \) with \( e \in E \), then \( e \) is said to be active at state \( x \), \( \delta(x, \lambda) = x' \) means that the system will move from \( x \) to \( x' \) according to the “always occurring” event \( \lambda \), i.e., without waiting for any external input symbol. In this case, \( x \) is said to be a \( \lambda \)-state. \( \lambda \)-transitions and \( \lambda \)-states are used to represent explicitly internal switches, for example high priority switches, that do not require any action from the controller and are directly generated by the system. Moreover, when a \( \lambda \)-transition (i.e.,

![Fig. 2: Example of labeled finite state automaton with inputs](image2)
a transition associated with event \( \lambda \) exists from a given state \( x \) then it will be the single transition outgoing from \( x \), i.e., \( \sum_{e \in E} |\delta(x, e)| = 1 \). Note that multiple labels (a subset of \( Q \)) could be provided by one transition and \( \varepsilon \) is used when no label is generated.

**Example 1**: Consider the LFAI of Figure 2. It is defined by a set of 8 states \( X = \{x_0, \ldots, x_7\} \) (with three \( \lambda \)-states \( x_3, x_5, x_7 \) represented by dashed circles) where \( x_0 \) is the initial state. The set of symbols is \( E = \{a, b\} \) and the set of labels is \( Q = \{A, B, C\} \). The notation "\( a : A \)" means that the system switches from state \( x_0 \) to state \( x_1 \) when it receives symbol \( a \) and that this change delivers label \( A \). Thus, it holds \( \delta(x_0, a) = x_1 \) and \( \text{Obs}(x_0, a) = A \).

A control sequence of length \( n \) sent by a controller is denoted by \( i = e_1 \ldots e_n \) with \( e_h \in E, h = 1 \ldots n \). It is completed by the \( \lambda \) events that are generated spontaneously by the system leading to the corresponding executed sequence \( i' = e_1' \ldots e_m' \), \( e_h' \in E_\lambda, h = 1 \ldots m \) with \( m \geq n \). Due to the determinism of the considered formalism, a single \( i' \) is associated to a given \( i \).

We introduce \( \delta^* \) and \( \text{Obs}^* \) as the trivial extensions of \( \delta \) and \( \text{Obs} \) functions defined recursively, for an executed sequence \( i' \), by \( \delta^*(x, e_i') = \delta(\delta(x, e_i'), i') \) and \( \text{Obs}^*(x, e_i') = \text{Obs}^{\delta}(\delta(x, e_i'), i') \). A trajectory, denoted \( \sigma(x, i') \), of \( m + 1 \) successive states could be obtained from \( x \) as:

\[
x_{j_0} \xrightarrow{e_1' \text{Obs}(x_{j_0}, e_1')} x_{j_1} \ldots x_{j_m-1} \xrightarrow{e_m' \text{Obs}(x_{j_m-1}, e_m')} x_{j_m}
\]

where \( x_{j_0} = x \) and \( x_{j_m} = \delta^*(x, i') \). The sequence of sets of labels generated by \( i' \) is denoted as \( \sigma = \text{Obs}^*(x, i') = \text{Obs}(x_{j_0}, e_1') \ldots \text{Obs}(x_{j_m-1}, e_m') \). We use \( (x_{j_{m-1}}, e_m') \in \sigma(x, i') \) to refer to a transition from the state \( x_{j_{m-1}} \) driven by the symbol \( e_m' \) in the trajectory \( \sigma(x, i') \).

Finally, when an attacker inserts orders or manipulates the control sequence \( i \) sent by the controller, the resulting sequence is called an attack sequence which is denoted as \( i_{\text{a}} \). Its corresponding executed sequence \( i'_{\text{a}} \) is called an executed attack sequence and the generated sequence of sets of labels to be erased is denoted as \( \sigma_{\text{a}} \).

### III. UNCERTAIN COST GRAPH

In this section, we consider that the attacker knows the model of the system, and can manipulate the symbol and label events. In particular, we consider moving attacks that are composed by sequences of symbols generated exclusively by the attacker, i.e., \( i_{\text{a}} = i \), and sequences of sets of outputs erased by the attacker. An insertion cost \( c_I \) is defined for each symbol and a deletion cost \( c_D \) is defined for each label. Note that \( c_I(\lambda) = 0 \) and \( c_D(e) \) is either defined as single values or as intervals when some uncertainties exist about such costs. Uncertainties may exist for various reasons, including the risk (from the attacker’s viewpoint) that the controller performs some actions during an attack.

In our previous work [15], fixed values of costs have been considered and an Adding Control Graph (ACG) has been developed. The objective was to characterize the attack sequences of minimal cost. The nodes of an ACG are those of the LFAI while each edge corresponds to a transition of the LFAI with a weight given as follows. Let \( x, x' \in X \) be two states in the LFAI and \( e \in E_\lambda \) such that \( \delta(x, e) = x' \). The weight \( c_{\text{ACG}}(x, e) \) of the arc corresponding to transition \( \delta(x, e) = x' \), is given by:

\[
c_{\text{ACG}}(x, e) = c_I(e) + \sum_{q \in \text{Obs}(x, e)} c_E(q) \tag{2}
\]

Now, let us introduce the Uncertain Adding Control Graph (UACG) that is similar to the ACG except that the weight of each edge is a positive interval. For this purpose, let us first define for each symbol \( e \in E \) the insertion cost interval as:

\[
C_I(e) = [c_{I}^{-}(e), c_{I}^{+}(e)],
\]

and for each label \( q \in Q \), the deletion cost interval as:

\[
C_E(q) = [c_{E}^{-}(q), c_{E}^{+}(q)].
\]

Note that \( C_I(\lambda) = C_E(e) = [0, 0] \). Let \( x, x' \in X \) be two states in the LFAI and \( e \in E_\lambda \), such that \( \delta(x, e) = x' \). The weight \( c_{U\text{ACG}}(x, e) \) of the edge corresponding to transition \( \delta(x, e) = x' \) is associated to the interval \( c_{U\text{ACG}}(x, e) = [c_{U\text{ACG}}^{-}(x, e), c_{U\text{ACG}}^{+}(x, e)] \),

\[
\begin{align*}
& c_{U\text{ACG}}^{-}(x, e) = c_{I}^{-}(e) + \sum_{q \in \text{Obs}(x, e)} c_{E}^{-}(q), \\
& c_{U\text{ACG}}^{+}(x, e) = c_{I}^{+}(e) + \sum_{q \in \text{Obs}(x, e)} c_{E}^{+}(q).
\end{align*}
\]

**Example 2**: Let us consider that the costs to insert or delete each symbol and label for the LFAI of Figure 2 are given by \( c_I(a) = 2, c_I(b) = 1, c_E(A) = 1, c_E(B) = 0, c_E(C) = 1 \). From these values, the cost of each transition of the corresponding ACG could be computed for the "certain case" as reported in Figure 3 (a). For instance, the cost to drive the system from state \( x_0 \) to state \( x_1 \) is equal to 3 which corresponds to the sum of the cost to insert the symbol \( a \) and the cost to erase the label \( A \) that results from the state switch, i.e., \( c_{ACG}(x_0, a) = c_{I}(a) + c_{D}(A) = 2 + 1 = 3 \).

Concerning the "uncertain case" and considering the cost intervals, \( C_I(a) = [2, 3], C_I(b) = [1, 2], C_E(A) = [0, 1] \).

Intervals are manipulated according to the IEEE 1788 standard for interval arithmetic [18]. In particular, \([a, b] + [c, d] = [a + c, b + d]\).
For instance, the cost interval to drive the system from state $x_0$ to state $x_1$ is equal to $C_{UACG}(x_0, a) = [c_F^((a)), c_E^((a)) + c_E^((A))] = [3, 7]$.

IV. ANALYSIS OF STEALTHY ATTACKS

In this section, we consider stealthy moving attacks that aim to drive the system from the current state to a state from a subset $\mathcal{F} \subseteq X$ of forbidden states. This set could include deadlocks or other dangerous states. To perform a stealthy moving attack, the attacker inserts an attack sequence (i.e., a wrong sequence of symbols) and, in the same time, it erases the observable traces (i.e., a sequence of sets of labels) that the attack sequence has generated. Uncertain costs, associated to the attacker’s actions, are used to take into account the variation, with respect to the minimal cost attack to state $x$.

Definition 2: Let $i_a = e_1 \ldots e_n$ be a stealthy moving sequence that drives the system from a state $x$ to a forbidden state in $\mathcal{F}$. The cost of $i_a$ is included in interval $c_{MA}(i_a, x)$ defined by

$$c_{MA}(i_a, x) = \sum_{h=1, \ldots, n} C_{UACG}(x_jh, e_h).$$

Since many possible attacks with variable costs exist to reach a forbidden state, two particular attacks are considered: the attack of minimal cost and the attack of maximal robustness. The search of such attacks is motivated by the assumption that the attacker has a limited credit to perform its attack.

Definition 3: An attack sequence $i_a^*$ of minimal cost $c^*$ is an attack that drives the system from a state $x$ to a forbidden state in $\mathcal{F}$ such that other attacks from $x$ to $\mathcal{F}$ have a cost at least equal to $c^*$.

Observe that the attack $i_a^*$ corresponds to the worst case from the controller perspective since the attacker could reach $\mathcal{F}$ from $x$ with the lowest cost.

Definition 4: An attack sequence $i_a^*$ of maximal robustness (or robust attack) is an attack that drives the system from a state $x$ to a forbidden state in $\mathcal{F}$ such that the possible cost variation, with respect to the minimal cost attack to $\mathcal{F}$, is minimal.

A. Moving attack of minimal cost

In order to compute the cost of a stealthy moving attack in an UACG, the notion of scenario is first introduced [16].

Definition 5: A scenario $S$ is an assignment of a single value $c_{UACG}(x, e, S) \in C_{UACG}(x, e)$ for each state $x \in X$ and event $e \in E$ active in $x$.

We refer to $ACG(S)$ as to the adding control graph obtained from the UACG associated to scenario $S$. The cost of the stealthy moving attack from $x$ to $x_f \in \mathcal{F}$ in scenario $S$ that inserts $i_a$ at $x$ and erases the generated sequence of sets of labels $o_a$ is obtained according to its corresponding executed attack sequence $i'_a$ and the resulting trajectory $\sigma(x, i'_a)$ of the form (1) that ends at $x_f$. It is named as the cost of attack sequence $i_a$ at state $x$ in scenario $S$ and computed as:

$$c_{MA}(x, i_a, S) = \sum_{(x', e) \in \sigma(x, i'_a)} C_{UACG}(x', e, S).$$

We denote by $S$ the set of possible scenarios. Observe that it remains non tractable to enumerate all possible scenarios.

Proposition 1: The attack sequence $i_a^*$ of minimal cost from state $x \in X$ to a forbidden state in $\mathcal{F}$ is defined by

$$i_a^* = \arg \min_{i_a \in I_a(x, \mathcal{F})} \{c_{MA}(x, i_a, S_{min})\},$$

where $I_a(x, \mathcal{F})$ is the set of attacks that move the state from $x$ to any state in $\mathcal{F}$ and $S_{min}$ is the scenario of minimal costs in which $c_{UACG}(x', e, S) = c_{UACG}(x', e)$ for each state $x' \in X$ and event $e \in E$ active in $x'$.

Proof: Let us first consider a given scenario $S \in S$. The stealthy moving attack of minimal cost from state $x$ to a given state $x_f \in \mathcal{F}$ in scenario $S$ can be obtained by using the well-known Dijkstra algorithm in $ACG(S)$ and the cost $c_{MA}(x, x_f, S)$ of such attack satisfies:

$$c_{MA}^*(x, x_f, S) = \min_{i_a \in I_a(x, x_f)} \{c_{MA}(x, i_a, S)\},$$

where $I_a(x, x_f)$ is the set of attacks that move the state from $x$ to $x_f$. Now by repeating the use of Dijkstra algorithm to each state $x_f \in \mathcal{F}$, one can compute the minimal cost $c_{MA}^*(x, \mathcal{F}, S)$ from current state $x$ to any state $x_f \in \mathcal{F}$ in scenario $S$:

$$c_{MA}^*(x, \mathcal{F}, S) = \min_{x_f \in \mathcal{F}} \{c_{MA}^*(x, x_f, S)\} = \min_{i_a \in I_a(x, \mathcal{F})} \{c_{MA}(x, i_a, S)\}.$$
B. Robust moving attack

In this section we are interested in determining robust attacks with respect to a set of known cost intervals that model the uncertainties. The criterion used here to classify an attack as robust or not is the maximal regret. This criterion initially proposed in the context of game theory was adapted to robust optimization in [19], [20], [21] and is extended here for defining robust attacks. Let us consider an UACG and first introduce the notion of regret.

Definition 6: The regret $r_{MA}(x, f, i_a, S)$ of a moving attack $i_a$ from a state $x$ to a state $x_f \in F$ in a given scenario $S$ is the difference between the costs of the attack sequence $i_a$ and the one of minimal cost to reach $x_f$ in scenario $S$:

$$r_{MA}(x, x_f, i_a, S) = c_{MA}(x, i_a, S) - c_{MA}^{\ast}(x, x_f, S). \quad (8)$$

The regret reflects the additional cost that the attacker may spend to reach $x_f$ with respect to the attack of minimal cost. The notion of regret can be extended to a set of forbidden states $F$ for $i_a$ in scenario $S$ by:

$$r_{MA}(x, F, i_a, S) = c_{MA}(x, i_a, S) - c_{MA}^{\ast}(x, F, S). \quad (9)$$

Proposition 2: The attack sequence $i_a^{\ast}$ of maximal robustness from state $x \in X$ to a forbidden state in $F$ is given by

$$i_a^{\ast} = \arg \min_{i_a \in I_a(x, F)} \{r_{MA}(x, F, i_a, S_{\max}(i_a))\} \quad (10)$$

with $S_{\max}(i_a)$ the scenario where the cost of each transition $(x', e) \in \sigma(x', i_a^{\ast})$ is the executed attack sequence that corresponds to $i_a$ is assumed to take its maximal value $c_{UACG}^{\ast}(x', e)$, whereas the costs of all other transitions in the UACG take their minimal values.

Proof: The regret $r_{MA}(x, x_f, i_a, S)$ of an attack sequence $i_a$ from $x$ to a given state $x_f \in F$ reaches its maximal value for the particular scenario $S_{\max}(i_a)$ [16]:

$$r_{MA}(x, x_f, i_a, S_{\max}(i_a)) = \max_{i_a \in S} \{r_{MA}(x, x_f, i_a, S)\}.$$  

It can be extended to a set of forbidden states $F$:

$$r_{MA}(x, F, i_a, S_{\max}(i_a)) = \max_{i_a \in S} \{r_{MA}(x, F, i_a, S)\}.$$  

The stealthy moving attack of maximal robustness from $x$ to $F$ corresponding to attack sequence $i_a^{\ast}$ is the attack with the minimal value of maximal regret. $i_a^{\ast}$ is obtained in three steps:

- compute the set $I_a(x, F)$ of attacks that correspond to non cycling trajectories from $x$ to $F$ in the LFAI (it is not necessary to consider the trajectories with one or more cycles because such trajectories include an additional cost and regret),
- for each attack $i_a$ from $x$ to $F$, compute the maximal regret $r_{MA}(x, F, i_a, S_{\max}(i_a))$,
- compute the minimal value $r_{MA}^{\ast}(x, F)$ of the maximal regret of the attacks from $x$ to $F$:

$$r_{MA}^{\ast}(x, F) = \min_{i_a \in I_a(x, F)} \{r_{MA}(x, F, i_a, S_{\max}(i_a))\}.$$  

Consequently, the stealthy moving attack from state $x$ to $F$ of maximal robustness is obtained according to the attack sequence of Equation (10) and Proposition 2 holds. Note that, in general, $i_a^{\ast}$ does not coincide with $i_a^{\ast\ast}$.

Remark 1: the time complexity to determine the robust attack that drives the system from state $x$ to a state in $F$ is $O(|I_a(x, F)||V| \cdot \log |X|)$. It depends on the number of non-cycling trajectories $|I_a(x, F)|$ which can be bounded by $|F| \times \sum_{k=0}^{n} \frac{n}{(n+k)}$ with $n = |X| - (|F| + 1)$ and on the complexity of Dijkstra’s algorithm given by $O(|V| \cdot \log |X|)$ where $|V|$ is the number of UACG edges.

Example 3: Consider again the LFAI of Figure 2 and an attack that aims to change the current state and mask the traces generated by the attack. Specifically, the attacker aims to drive the plant from state $x_0$ to state $x_6$. There exist several attack sequences to reach $x_6$, in particular the sequences of symbols, $ab(ab)^{\ast}b$ and $ba$. According to Dijkstra algorithm applied on ACG in Figure 3(a), the trajectory of minimal cost is $x_0 \rightarrow x_2 \rightarrow x_3 \rightarrow x_6$. Thus, the attacker needs to insert symbol $b$, next symbol $a$, and erase labels $A$ three times and next $B$ once. The attack sequence of minimal cost is then given by $i_{a}^{\ast} = ba$ which corresponds to the executed attack sequence $(i_{a}^{\ast})^{\ast} = ba\lambda$.

Consider now the UACG of Figure 3(b) that extends the path $x_0 \rightarrow x_6$. Let us first remark that $ACG(S_{\min})$ associated to this UACG is the one represented in Figure 3(a). Thus $c_{MA}^{\ast}(x_0, x_6, S_{\min}) = 6$ and $i_{a}^{\ast} = ba$. For the robust attack that drives the plant from state $x_0$ to state $x_6$, observe that only two attack sequences that correspond to non-cycling trajectories exist to move the system from $x_0$ to $x_6$ given by $I_a(x_0, x_6) = \{abb, ba\}$. The maximal regret of the two attacks are respectively $r_{MA}(x_0, x_6, abb, S_{\min}(abb)) = 7 + 4 + 6 - 2 - 3 - 1 = 11$ whereas $r_{MA}(x_0, x_6, ba, S_{\min}(ba)) = 6 + 7 + 6 - 3 - 2 = 12$. To conclude, $r_{MA}^{\ast}(x_0, x_6) = 11$ and thus the robust attack sequence is given by $i_{a}^{\ast} = abb$. Note that the attack of minimal cost is different from the robust one.

Remark 2: The proposed approach, dedicated to CPS, could be enlarged to any system provided that its associated model explicitly defines the states space and the input/output information.

V. CASE STUDY

We consider for this case study a part of an industrial plant (see Figure 4) composed by two production lines and two robots of different types (R1, R2, and R3). Each robot executes the operation of moving products from one place to another. The plant is modeled as an automaton with two inputs (robot operation commands) and one output (configuration of the plant).

A. Description

The considered system is represented by the output synchronized Petri net (OutSynPN) given in Figure 5 (see [17] for more details on this formalism). Places $p_1$ and $p_5$ represent,
physical system provides some feedback to the controller as resumed in Table II, where \( \uparrow m_i \) and \( \downarrow m_i \) represent, respectively, an increasing or decreasing of the marking of place \( p_i \) (\( m_i = M(p_i) \)) and MA stands for mobile agent.

Note that if we consider two robots R1, one robot R2 and two robots R3, noted as configuration 2R1/1R2/2R3, with two mobile agents per production line, the physical system is blocked in two situations: robot R2 is used by line 1 and both robots R3 are used by line 2 ; both robots R1 are used by line 1 and robot R2 is used by line 2. These both situations are detected by the feedback \( E \) provided by the physical plant to the controller.

### B. Results

Let us now consider the costs associated with an insertion of symbol or a deletion of label by the attacker given by Table III. At the initial state, in configuration 2R1/1R2/2R3 with two mobile agents per production line, all robots are available and all mobile agents are located at the electric charging station. From the Petri net model, the LFAI could be determined and the uncertain adding control graph is obtained. This UACG, shown in Figure 6, is composed of 41 states where \( x_0 \) is the initial state and, \( x_{18} \) and \( x_{19} \) are deadlocks corresponding to the two blocked situations previously described.

We consider the set of forbidden states given by \( \mathcal{F} = \{x_{13}, x_{14}, x_{18}, x_{19}\} \) that correspond to the two deadlocks and two additional dangerous states from which the system cannot come back to a normal state.

The attacker wants to determine an attack, weakly impacted by costs uncertainties, that drives the system from the initial state \( x_0 \) to one state of \( \mathcal{F} \). According to the approach proposed in this paper, the robust attack \( \gamma^*_a \) is computed by evaluating the maximal regret from \( x_0 \) to states \( x_{13}, x_{14}, x_{18} \) and \( x_{19} \) (see Table IV). Thus, according to Proposition 2, the robust attack \( \gamma^*_a = aba \) is obtained which corresponds to the smallest

### Table I: Control inputs

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Application</th>
<th>Control input</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>mobile agent of line 1</td>
<td>enter in line 1 &amp; move to area R1</td>
</tr>
<tr>
<td>b</td>
<td>mobile agent of line 2</td>
<td>enter in line 2 &amp; move to area R3</td>
</tr>
<tr>
<td>c</td>
<td>mobile agent of both lines</td>
<td>evacuate the line (i.e., the product) and move to the station</td>
</tr>
<tr>
<td>d</td>
<td>mobile agent of line 2</td>
<td>move to area R1</td>
</tr>
<tr>
<td>e</td>
<td>mobile agent of line 1</td>
<td>move to area R2 and/or area R3</td>
</tr>
</tbody>
</table>

### Table II: Plant output

<table>
<thead>
<tr>
<th>Label</th>
<th>Definition</th>
<th>Plant output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>( \uparrow m_2 )</td>
<td>a MA enters in line 1</td>
</tr>
<tr>
<td>B</td>
<td>( \uparrow m_6 )</td>
<td>a MA enters in line 2</td>
</tr>
<tr>
<td>C</td>
<td>( \downarrow m_4 )</td>
<td>a MA leaves line 1</td>
</tr>
<tr>
<td>D</td>
<td>( \downarrow m_8 )</td>
<td>a MA leaves line 2</td>
</tr>
<tr>
<td>E</td>
<td>((\downarrow m_{10} \land (m_2 = 2))\lor(\downarrow m_9 \land (m_2 = 2)\land (m_1 = 1))\lor(\downarrow m_{11} \land (m_6 = 2)\land (m_3 = 1)))</td>
<td>a blocked situation</td>
</tr>
</tbody>
</table>

### Table III: Insertion and deletion costs

<table>
<thead>
<tr>
<th>Symbol</th>
<th>( C_I )</th>
<th>( C_E )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>[2, 3]</td>
<td>[1, 4]</td>
</tr>
<tr>
<td>b</td>
<td>[1, 2]</td>
<td>[0, 2]</td>
</tr>
<tr>
<td>c</td>
<td>[1, 2]</td>
<td>[1, 2]</td>
</tr>
<tr>
<td>d</td>
<td>[2, 3]</td>
<td>[1, 3]</td>
</tr>
<tr>
<td>e</td>
<td>[1, 5]</td>
<td>[3, 4]</td>
</tr>
</tbody>
</table>
TABLE IV: Minimal costs and min-max regret

<table>
<thead>
<tr>
<th>State</th>
<th>Minimal cost</th>
<th>Min-max regret</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{13}$</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>$x_{14}$</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>$x_{18}$</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>$x_{19}$</td>
<td>11</td>
<td>15</td>
</tr>
</tbody>
</table>

regret $r_{M,A}^*(x_0, F) = 11$. The detail of the trajectory is as follow: $x_0 \xrightarrow{a:A} x_{14} \xrightarrow{b:B} x_8 \xrightarrow{a:AE} x_{13}$. Due to uncertainties in the insertion and deletion costs (see Table III), the actual cost of the attack $aba$ may vary in range $[11, 21]$. In the worst case, the cost may increase up to 21.

In order to point out the importance of computing the robust attack for both attacker and defender, consider also the attack $i_5^* = aebb$ of minimal cost $c_{M,A}^*(x_0, F) = 9$ that differs from $i_5 = aba$. It corresponds to the trajectory $x_0 \xrightarrow{a:A} x_1 \xrightarrow{e} x_3 \xrightarrow{b:B} x_7 \xrightarrow{b:BE} x_{14}$. When uncertainties exist, the cost of $aebb$ may vary within $[9, 23]$ and one can observe that, in the worst case the cost of $aebb$ exceeds 21 and that input sequence $aba$ becomes more performant than input sequence $aebb$.

C. Computation time evaluation

To evaluate our approach, a set of simulation has been run using Matlab® on an Intel® Core™ i7-8650U 2.11GHz CPU with 16Go RAM memory. First of all, the time needed to compute the previously robust attack, $i^*_5 = aba$, is 286.55 seconds. A simulation campaign has been conducted with different robots configurations, number of mobile agents and, cost intervals.

Table V shows the impact of the number of mobile agents and robots on the size of the UACG and on the computation time. Note that for the first four configurations, the computation time is bounded when the number of mobile agents per line is equal to the total number of robots. In fact, the size of the UACG, and consequently the number of attack trajectories, remain constant. This result can be generalised to all configurations as it is a structural property of the system’s OutSynPN model. The number of attack trajectories, $|I_a(x, F)|$, increases exponentially as the size of the UACG (represented by the number of nodes $|X|$ and the number of edges $|V|$) grows. As a consequence, the computational time becomes large (out of time (o.t.) is used when the computational time exceeds 3 hours).

The results of Table VI are obtained for a fixed robots configuration $2R1/1R2/2R2$ and two mobile agents in each production line. Firstly, it is shown how the number of attack trajectories (and thus the computation time) evolves with respect to the number of forbidden states. Secondly, we consider a variation of the cost intervals. The values, based on Table III, are given by $C_I = [c_I, c_I^+ + \Delta]$; $C_E = [c_E, c_E^+ + \Delta]$ where $\Delta$ is the parameter. It is important to point out that the variation of the cost intervals does not impact the computation time (insignificant time variation). This result is not surprising as the proposed approach does not consider an exhaustive investigation of all scenarios but considers only extremum scenarios that use minimal and maximal cost bounds.

TABLE V: Simulation results

| Robots config. | Agents per line | $|X|$ | $|V|$ | $|F|$ | $I_a(x, F)$ | Computation time (sec.) |
|---------------|----------------|------|------|------|-------------|-------------------------|
| Config.1:     | 1              | 10   | 2    | 3    | 0.016       |
| 1R1           |                |      |      |      |             |
| 1R2           | 3              | 17   | 2    | 7    | 0.035       |
| 1R3           | 4              | 17   | 2    | 7    | 0.025       |
| Config.2:     | 1              | 11   | 1    | 6    | 0.026       |
| 2R1           | 2              | 28   | 5    | 2    | 332         | 1.546                   |
| 2R2           | 3              | 36   | 7    | 4    | 1282        | 8.674                   |
| 2R3           | 4              | 38   | 8    | 2    | 2389        | 21.762                  |
| Config.3:     | 1              | 11   | 1    | 6    | 0.026       |
| 1R1           | 2              | 25   | 4    | 2    | 37          | 0.182                   |
| 1R2           | 3              | 36   | 6    | 3    | 145         | 1.071                   |
| 1R3           | 4              | 38   | 7    | 3    | 201         | 1.564                   |
| Config.4:     | 1              | 10   | 15   | 4    | 0.017       |
| 1R1           | 2              | 25   | 4    | 2    | 77          | 0.359                   |
| 1R2           | 3              | 33   | 6    | 2    | 175         | 1.046                   |
| 1R3           | 4              | 35   | 7    | 2    | 231         | 1.481                   |
| Config.5:     | 2              | 11   | 19   | 0    | –           |
| 2R1           | 2              | 33   | 7    | 3    | 1671        | 13.912                  |
| 2R2           | 3              | 51   | 11   | 2    | 31525       | 338.360                 |
| 2R3           | 4              | 69   | 15   | 3    | 566839      | 694.07                  |
| Config.6:     | 5              | 71   | 17   | 3    | 892942      | o.t.                    |
| 2R1           | 2              | 12   | 21   | 0    | –           |
| 2R2           | 3              | 66   | 15   | 2    | 2641486     | o.t.                    |
| 2R3           | 4              | 75   | 17   | 2    | 13299270    | o.t.                    |
| Config.7:     | 7              | 22   | 18   | 0    | –           |
| 2R1           | 2              | 89   | 9    | 1    | 16334       | 161.39                  |
| 2R2           | 3              | 146  | 15   | 3    | 4498087     | o.t.                    |
| 2R3           | 4              | 175  | 17   | 3    | 21598       | 279.02                  |

TABLE VI: Computation times with different sets $F$ and variable cost intervals

VI. CONCLUSIONS

This paper concerns vulnerability analysis of cyber-physical systems that include input and output events. By using a labeled finite automaton with inputs (LFAI) and an uncertain weighted graph that encodes the cost of the malicious actions, we propose an approach to evaluate the robustness of the attack and the vulnerability of the controlled system. This analysis can be used to isolate the weaknesses of the system (in terms of cyber-security) such that improvements can be considered to compensate the vulnerabilities. This approach is illustrated in the case of moving attacks that aim to drive the system’s current state into a set of forbidden states. Limitations of the proposed approach lie in the difficulty to estimate the values of the cost intervals for real system, but also in its complexity related to the size of the graph that may grow exponentially.

The solutions obtained by robust optimization techniques such as the min-max regret could be too conservative [22]. Re-
placing the intervals by probabilistic distributions and connecting the probabilistic aspects to the costs leads to several open questions, such as the use of less conservative approaches, that will be considered in our future works. In addition, we will consider more general attack scenarios, including partial controllability and observability properties from the perspective of the attacker. We will also consider situations where the controller and attacker interact like a two-player game. Finally, we will explore situations where the attacker wants to freeze the state of the system in some particular situations.

REFERENCES


