Switched-based resilient control of cyber-physical systems

Juan Nolazco Flores, Vinh Hoa La, Ana Rosa Cavalli, Raul Ramirez Velarde, Raul Armando Fuentes Samaniego

Switched-Based Resilient Control of Cyber-Physical Systems

MARIANA SEGOVIA-FERREIRA, (Student Member, IEEE), JOSE RUBIO-HERNAN, (Member, IEEE), ANA ROSA CAVALLI, (Member, IEEE), AND JOAQUIN GARCIA-ALFARO, (Senior Member, IEEE)
Institut Polytechnique de Paris, Télécom SudParis, 91120 Palaiseau, France

Corresponding author: Joaquin Garcia-Alfaro (garcia_a@telecom-sudparis.eu)

This work was supported in part by the Cyber CNI Chair of the Institut Mines-Télécom, and in part by the European Commission through the H2020 SPARTA Project, under Grant 830892.

ABSTRACT We present a control-theoretic approach to achieve Cyber-Physical Systems (CPS) resilient designs. We assume situations in which the CPS must maintain the correct operation of a set of crucial functionalities despite ongoing adversarial misbehavior. The approach is based on a moving target defense paradigm, driven by a linear switching of state-space matrices, and applied at both the physical and network layers of a networked-control system. We show that the final system maintains stability. We also evaluate, via simulation, a step-by-step procedure that takes a transfer function, representing the dynamics of the physical process. As a result, we obtain a resilient CPS design structured around a topology of decentralized controllers. Results show that the obtained approach is both innovative and promising.

INDEX TERMS Cyber-physical security, infrastructure security, cyber resilience, moving target defense, switched linear systems, networked control systems.

I. INTRODUCTION

A typical CPS architecture is composed of a multitude of physically and functionally heterogeneous components that work in a distributed networked-based manner [1]. The controllers coordinate the action of the system in a distributed way by receiving information and sending commands. Resilience is especially relevant in these systems, since attacks can manifest significant physical effects [2]. For example, cyber-physical attacks may lead to negative impact on human safety, cause harm in natural environments, interrupt industrial process continuity; leading to large economic losses, generating legal problems and damaging the reputation of the affected organizations.

On the other hand, the CPS complexity and heterogeneity of their components introduced significant difficulties to secure them. In addition, CPS attacks are difficult to trace, classify or identify the original threat, which may move or spread, and target multiple components of the system. For this reason, research on cyber attacks and secure control has found increasing interest [3].

Skilled adversaries may be undetectable if they learn the model of the underlying physical process as showed in Teixeira *et al.* [4], [5], who discusses the security of control systems based on the knowledge that the adversary has about the system. Then, it is presented the mathematical model for different adversary models such as zero-dynamics attack, covert attack, false data injection attack and replay attack. In addition, Sanchez *et al.* [3] provides a detailed bibliographic review of cyber-physical attacks given illustrative examples.

On the other hand, resilience can be defined as the ability to resist, absorb, recover or successfully adapt to adversity or a change in conditions [6]. The implementation of resilience methods aims to ensure essential operations, reducing potential damages, maintaining critical functions level and rapid recovery. Resilient CPS are expected to keep an acceptable performance, even in the case of faults, disruptions or attacks. This refers to the ability to ensure that system outputs are correct, within acceptable operating thresholds and the normal operation can be restored despite local faults or attacks.

Assuring that a system is resilient to cyber-physical attacks is a non-trivial task, resilience-by-design approaches assume the incorporation of resilience against such attacks since the initial conception of the system.

As showed in recent surveys [7]–[9], most of the existing approaches require adding extra hardware [10], which may be expensive. Other solutions use detection approaches with recovery strategies [11]–[14] that usually use state estimation to maintain an understanding of the system state under attack, even when a subset of inputs and outputs are compromised. These techniques work as traditional detection and mitigation approaches but do not provide resilience-by-design or prevent the execution of malicious commands. Having a reliable estimate allows a defender to better understand the portions of a system that have been compromised and design attack specific solutions to counter the adversary actions. In addition, these approaches require to include also a mechanism to ensure the correct feedback control after detecting the attack.

Another strategy is to use game-theoretic approaches such as [15] and [16]. These approaches have the advantage of providing proactive dynamic defenses. However, they consider only cyber aspects without the physical part. Considering control theory aspects in an approach to face cyber-physical adversaries is vital to ensure the stability of the process that is coupled with the software components.

In this article, we focus on resilience via MTD approaches [17], using physical model mutations and network reconfiguration. The proposed approach builds a resilient-by-design system using a switched control. This technique allows the system to change its design periodically by self-heal. Our main contributions are as follows: (1) an approach to build resilient cyber-physical systems capable of ensuring close-loop stability in presence of cyber-physical adversaries and (2) an experimental work that validates the approach via simulation. The approach is innovative since it does not require a detection and reaction mechanism as in the existing literature. The system has the capability of self-healing due to its design, using a collaborative control system with mutating control laws. The network and physical process controllers collaborate to improve the resilience of the system.

Paper organization — Section II establishes the preliminaries. Section III surveys related work. Section IV presents the approach. Section V provides experimental validation and effectiveness discussion. Section VI concludes the paper.

II. PRELIMINARIES

A. CONTROL-THEORETIC ASSUMPTIONS

Cyber-Physical Systems (CPS) can be theoretically modeled using representations that relate to each possible input signal, the corresponding output signal. The two main mathematical approaches to model this are the *transfer function* and the *state-space model*. Both representations are equivalent since they are based on the differential equations that model the behavior of the physical process being controlled. For this reason, it is possible to transform one representation into the other and vice-versa.

Normally, a CPS design process starts with the transfer function since it is the most direct form starting from the differential equations of the process. The transfer function $G(s)$ is the ratio of the Laplace transformation using the complex variable s of the output $Y(s)$ to that of the input $U(s)$. It is represented as showed in Equation (1) by the division of two polynomials, the numerator is created by taking the coefficients b_i of the output differential equation and the denominator using the coefficients a_i of the input differential equation.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{i=0}^{m} b_i s^{m-i}}{\sum_{i=0}^{n} a_i s^{n-i}} \tag{1}$$

A transfer function with multiple inputs and multiple outputs is usually represented in a matrix form which indicates the relationship of each input and each output of the system. Using well-known control theory techniques [18], it is possible to transform the transfer function into a state-space model by expressing the differential equations into matrices forms, cf. Equation (2):

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{2}$$

where $x_k \in \mathbb{R}^n$ is the vector of the state variables at the k-th time step, $u_k \in \mathbb{R}^p$ is the control signal and $w_k \in \mathbb{R}^n$ is the process noise that is assumed to be a zero-mean Gaussian white noise with covariance Q, *i.e.* $w_k \sim N(0, Q)$. Moreover, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times p}$ are respectively the *state* matrix and the *input* matrix. The value of the output vector $y_k \in \mathbb{R}^m$ represents the measurements produced by the sensors that are affected by a noise v_k assumed as a zero-mean Gaussian white noise with covariance R, *i.e.* $v_k \sim N(0, R)$ and $C \in \mathbb{R}^{m\times n}$ is the output matrix that maps the state x_k to the system output.

One interesting property is that the system models are not unique. On the contrary, a system has many equivalent representations that can be obtained using matrix factorization techniques. In addition, in multivariable CPS design, the rule is to use a decentralized control system consisting of independent controllers due to its many advantages: simplified design and tuning, flexibility, failure tolerance, among others [19]. As a result, the overall system[1] is no longer controlled by a single controller but by several independent controllers. In consequence, it is necessary the partition of the system into manageable sub-problems and to assume that the controllers know only a part of the overall information of the system. All this allows having many possible implementations for the same system, which can be used to implement a Moving Target Defense (MTD) mechanism to improve the system resilience.

Traditionally, CPS remains unchanged during long periods. For this reason, they become vulnerable to adversaries who can gather data and use their precise knowledge of the system dynamics, communications and control to damage

[1] Often referred to as *plant* in the related literature.

the system. MTD has emerged as a strategy to add uncertainty about the state and execution of a system to prevent in a proactive and reactive mode the insider adversaries [17].

In critical CPS, there is a control system that takes action over the physical process and a safety system that reacts to shut down in a safe way when the control system is not working properly. For safety reasons, it is not advisable to create reaction or mitigation mechanisms that may interfere with this safety shut down. The proposed approach allows the system to heal itself by design without any addition detection or reaction mechanism that identifies or mitigates threats rather than the traditional safety system.

B. NETWORKING AND ADVERSARIAL ASSUMPTIONS

We assume realizable networked systems, whose elements are proper, causal and stable. We assume infrastructure environments that are connected using programmable networks via, e.g., Software Defined Network (SDN) technologies [20]. We also assume that there is a secure management of SDN controllers and switches, to synchronize operational and security parameters [21].

We consider discrete linear time-invariant (LTI) modelling of the physical processes. Notice that the physical process does not need to be necessarily linear. Non-linear physical processes are usually linearized using well-known techniques in the literature. Such techniques are considered out of the scope of this article. Likewise, previous work has already shown that, from a security standpoint, adversaries can attack and hide better when systems are modelled as LTI. For instance, since the degree of the polynomial description associated to the physical process is usually higher when systems are modelled as LTI, the number of points available to an adversary to attack and hide are also higher (cf. [13] and citations thereof for further details). Hence, it is assumed that security solutions that are valid under the LTI assumption, are also valid under non-LTI assumptions, since the non-LTI case is less favorable to the adversary. In other words, by addressing the LTI case, our work tackles the less favorable case for security, rather than the easiest case for the defender.

The objective of the adversary is to cause a malfunction in the system by modifying the network traffic and affecting the control system. To achieve this, the adversary corrupts the system inputs and outputs that are sent using the data network. In particular, the most powerful adversary is the parametric cyber-physical adversary mentioned in [13] because of the ability to estimate the system parameters, i.e., the adversary knows the system dynamics. For example, using techniques such as machine learning, ARX (*autoregressive with exogenous input*) or ARMAX (*autoregressive-moving average with exogenous input*) models. The system model working under the effect of this adversary can be modeled mathematically as:

$$x'_{k+1} = Ax_k + Bu'_k \tag{3}$$

where u'_k represents an attack to the control input, i.e., in the commands sent from the controller to the actuators. In addition, this adversary can inject specific malicious measurements designed to deceive the control system:

$$y'_k = C'x_k \tag{4}$$

where C' represents an attacker that is able to create a sensor output y'_k that is correlated with the real u_k control input sent by the controller. This means, that the attacker is capable of sending a sensor output according to the system state x_k that the controller is expecting to receive. This attack is designed to mislead the system or destabilize its physical processes. The adversary aims at evading detection, by hiding the actions as faults or errors, whose random nature are much easier to be identified and corrected. The closer the matrices A, B and C that the adversary learned are to the real matrices in the controller, the more difficult is to detect the adversary.

Finally, the adversary is assumed to be placed in a remote location but gained access to the internal network exploiting cyber vulnerabilities. The adversary uses the network traffic to perform the attacks, as an insider. In addition, the adversary is able to change positions in the network. The adversary performs malicious actions in the data layer of the network domain. This means that the adversary is not attacking the SDN plane itself, e.g., the SDN control layer; but the data traffic that is flowing through the SDN network.

III. RELATED WORK

A static structure allows adversaries to collect information and perform long-term analysis. In addition, the uniformity of components allows attackers to expand the damage scope after they find one vulnerability. For this reason, MTD approaches provide strategies that change the system over time to increase its complexity, attack cost or limit the exposure of vulnerabilities [22]. The mechanisms are usually applied at the network or the node level [17]. Next, we summarize proposals for both levels as well as approaches specially designed for CPS.

A. NETWORK MTD APPROACHES

The *endpoint information* (such as MAC address, IP address, port, protocol or encryption algorithm) and the *forwarding path* (links and routing nodes) are two key elements in network transmission and it can be used to identify the source and destination nodes. Hence, it is important to protect this information as part of the attack surface.

Some approaches that protect the endpoint information are as follow. Antonatos *et al.* [23] propose the use of Network Address Space Randomization (NASR) to handle worm attacks. The method analyzes and discriminates the potentially infected endpoints and the nodes are forced to frequently change their IP address by using DHCP protocol. Al-Shaer *et al.* [24] proposed Random Host Mutation that assigns virtual IP addresses that change randomly and synchronously in a distributed way over time. To prevent disruption of active connections, the IP address mutation is managed by network appliances and totally transparent to the end-host. MacFarland and Shue [25] hide the endpoint MAC,

IP and port numbers by setting up DNS hopping controller and synthetic addressing information in place of the real one with the help of NAT rules. This can be considered to be chosen at random within certain validity constraints.

Other approaches provide protection of the forwarding path information, i.e., it randomly selects routing nodes to change the forwarding paths while ensuring reachability. For example, Dolev and David [26] uses a secret sharing technique to encrypt its data and create n shares, and only less than k parts can be allowed to transmit in the same path. In addition, to reconstruct the data, the destination needs to have at least k shares out of the n shares that were sent. The approach objective is to provide private and secure interconnection between the data centers. Aseeri *et al.* [27] proposes an approach to improve the diversity of forwarding paths to deal with eavesdropping attacks in the SDN data plane. It uses bidirectional multiple routing paths to reduce the severity of data leakage. The SDN controller applies the multipath mechanism both ways, from the sender side and the receiver side. By negotiating migrating paths between source and destination, the forwarding path is changed randomly during transmission. Duan *et al.* [28] proposed a Random Route Mutation technique that enables changing randomly the route of the multiple flows in a network simultaneously to defend against reconnaissance, eavesdrop and DoS attacks while preserving end-to-end QoS properties. Ma *et al.* [29] proposes an approach for self-adaptive end-point hopping, which is based on adversary strategy awareness and implemented using SDN. This method periodically changes the network configuration in use by communicating endpoints.

B. NODE MTD APPROACHES

Platform environment and software applications can be diversified to protect from adversaries. Diversity proposes to have many forms of the same object because this design can reduce the probability of intrusion [30]. Address space, instructions or data randomization are three typical ways to achieve platform environment diversification [31]. Another technique is software application isomerization that is a mechanism that changes codes dynamically to enhance the heterogeneity of software applications under the premise of ensuring functional equivalence. Depending on the application software life cycle, it can be divided into transformation mechanisms adopted during software compilation and link or transforming mechanism implemented during software load and execution [17]. Finally, programmable reflection is a meta-programming technique that has the potential to allow a programmable system to manipulate itself at runtime [32].

C. CPS-BASED MTD APPROACHES

Most CPS-based MTD approaches have been proposed to control adversaries situated in the end devices, i.e., actuators and sensors. For example, in [11], authors propose a MTD strategy that randomly changes the availability of the sensor data, so that it is harder for adversaries to achieve stealthy attacks. This approach uses switched control systems that allow to detect sensor compromise and to minimize the impact of false-data injection attacks. Griffioen *et al.* [33] proposes a MTD approach for recognizing and isolating CPS integrity attacks on a set of sensors and actuators by introducing stochastic time-varying parameters in the control system. The underlying random dynamics of the system limits the attacker's knowledge of the model. Weerakkody and Sinopoli [12] proposes a MTD approach to minimize identification in CPS, i.e., to limit the adversary's knowledge of the system model with the goal of identifying sensor attacks by changing the dynamics of the system as a function of time. Kanellopoulos and Vamvoudakis [10] proposes an approach to mitigate sensor and actuator attacks by formulating a control algorithm based on MTD that provides a proactive and reactive defense mechanism. It uses a stochastic switching structure to alter the parameters of the system and make it more difficult for the adversary to perform a system reconnaissance.

The existing network MTD approaches are mainly focused on common Internet applications and may not be suitable for CPS real-time applications. On the other hand, node MTD approaches are useful to face adversaries that target the platform or software running in a host. However, the main issue in CPS are adversaries that modify the network traffic. Finally, the proposed CPS-based MTD approaches aim at detecting or mitigating attacks but they do not offer a resilience solution that allows a system to self-heal from adversaries. In this context, we propose an approach that provides resilience-by-design that does not require any detection or mitigation mechanism to work since the system itself is capable of repairing the adversary damage caused by introducing malicious traffic. The proposed system design applies a distributed network and node MTD approach for CPS based on modifying the physical model of each node, i.e., modifying the transfer function that they execute in a coordinated manner that allows facing network adversaries while the globally distributed transfer function of the system remains unchanged.

D. SWITCHED SYSTEMS

Switching control techniques are based on changing between different controllers in the adaptive context while achieving stability. Many systems encountered in practice exhibit switching between several subsystems that is dependent on various environmental factors. Switched systems with all the subsystems described by linear differential equations are called switched linear systems.

The importance of such control methods also stems in part from the existence of systems that cannot be asymptotically stabilized by a single continuous feedback control law [34].

The control theory research community has studied how to ensure the stability of switched control under different systems' characteristics [35], [36]. For example using techniques for arbitrary switching [36], [37] [38], [39], such as Common Quadratic Lyapunov Functions and Switched Quadratic Lyapunov Functions; or restricted switching [40], [41] [42], [43] [36], [44] [39], such as slow switching

or dwell-time switching, Multiple Lyapunov Functions and Piecewise Quadratic Lyapunov Functions.

In an arbitrary switching, there are no restrictions to chose when to change controllers. However, a restricted switching may arise from natural physical constraints of the system. For example, in the automobile gear switching, there is a particular switching sequence to be followed; from first gear to the second gear, then third gear, etc. Another arbitrary switching condition may be, for instance, a certain bound on the time interval between two successive switchings.

IV. OUR APPROACH

In our approach, we propose to take as an input a CPS modeled by a transfer function and build a resilient equivalent system capable of controlling the same physical process using a Switched Linear Control System. This way, we design a Linear Time Variant (LTV) system. In particular, Switched Linear Systems have gained major attention in the control theory community in the last years since there exist unstable processes that are not possible to control with just one model but it is possible to design switched controllers for stabilizing it with piece-wise signals [34], [44]. In this article, we propose to use this control theory technique to improve the resilience of the process.

A switched system consists of a finite number of subsystems and a logical rule that orchestrates the switching between the subsystems. It may be modeled as follows:

$$x_{k+1} = f_{\sigma(k)}(x_k, u_k) \tag{5}$$

where $k \in \mathbb{Z}^+$ is the time interval, $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^p$ is the control input and σ is the logical rule that orchestrates the switching between the subsystems. It means that σ is a function $\sigma : \mathbb{Z}^+ \rightarrow \mathcal{I}$, where $\mathcal{I} = \{1, \ldots, N\}$ contains the indexes of the subsystems. A subsystem is determined by a pair $(\mathcal{M}_i,\ \mathcal{G}_i)$ where $\mathcal{M}_i = \{A_i, B_i, C_i : i \in \mathcal{I}\}$ is the set of physical system models and $\mathcal{G}_i = \{V_i, E_i : i \in \mathcal{I}\}$ is the set of graphs that represent the network connections in the CPS. Hence, σ define a piece-wise switching signal that is a time-varying definition of the process model and the network graph that is activated at time k. The physical model activated at time k is then defined by Equation (6) as follows:

$$\begin{aligned} x_{k+1} &= A_{\sigma(k)}x_k + B_{\sigma(k)}u_k \\ y_k &= C_{\sigma(k)}x_k \end{aligned} \tag{6}$$

whose system communicates through a network determined by the connectivity graph $\mathcal{G}_{\sigma(k)} = [V_{\sigma(k)}, E_{\sigma(k)}]$. The approach aims at protecting the system from network adversaries working at the node level by modifying the controller model and at the network layers modifying the endpoint information. In the sequel, we provide a procedure to build a resilient system.

Step 1 (Models Design): In this section, we analyze how to design the physical models in the subsystems, i.e., how to create the subset of matrices $\mathcal{M}_i = \{A_i, B_i, C_i : i \in \mathcal{I}\}$ that will be activated at each time period. There are two mechanisms to design equivalent control systems capable of controlling the same physical process. One possibility is to have redundant sensors and actuators, as proposed in [10]. This mechanism requires to add extra hardware to the system. So, the controller can choose at each time period which one to activate. The approach we propose is to design distributed controllers that modify in time the physical model they execute. The overall process is controlled by several independent controllers and altogether represent a decentralized controller, i.e., if at time k it is activated the control model with matrices A_i, B_i, C_i then there will be j controllers with $j \in 1 \ldots o$ and each controller will use a set of matrices A_{ij}, B_{ij}, C_{ij} where $A_i = \bigcup_{j=1}^{o} A_{ij}$, $B_i = \bigcup_{j=1}^{o} B_{ij}$ and $C_i = \bigcup_{j=1}^{o} C_{ij}$. Hence, the controllers have available only parts of the overall information.

In the sequel, we analyze how to derive the equivalent models starting from the initial transfer function as represented in Equation (1). The objective is to obtain different models expressed in the A_{ij}, B_{ij}, C_{ij} matrices which can be combined to represent the system dynamics as in Equation (2) and it allows deriving different sets of controllers capable of controlling the physical process.

Step 1.1: To obtain the equivalent representation we will factorize the matrices applying techniques similar to the ones used by the different approaches for decentralized control design [45], [46]. It consists of combining a diagonal controller $Q(s)$ with a block compensator $D(s)$ in such a way that the controller perceives the process dynamics $G(s)$ as a set of independent processes as showed in Equation (7):

$$G(s) \cdot D(s) = Q(s) \tag{7}$$

where $D(s)$ and $Q(s)$ are both $n \times n$ matrices of transfer functions, $Q(s)$ is diagonal and $D(s)$ invertible. Hence, the structure of the distributed controllers will be formed for n controllers executing the Q_{ii} transfer functions and each of these controllers is connected with n controllers executing the D_{ij} transfer function. In Figure 1(a), we show the structure for a 2×2 example.

To create this distributed design, the first step is to calculate $adjG(s)$ the adjudged matrix of G which is the transposition of the co-factor matrix of G.

Step 1.2: We build matrix $D(s)$ as follows. For each column $\hat{J} = \{1, .., N\}$, we select a row $\hat{I}$ to set that element $d_{\hat{I}\hat{J}}$ in the matrix $D(s)$ to unity. It is necessary to choose one for each column but not necessarily the diagonal ones.

After choosing the elements $(\hat{I}, \hat{J})$ to be set to one, the matrix $D(s)$ can be completed as follows:

$$d_{i\hat{J}} = \frac{adjG_{i\hat{J}}}{adjG_{\hat{I}\hat{J}}}$$

where $adjG_{i\hat{J}}$ is the $(i, \hat{J})$ element of $adjG(s)$ the adjugate matrix of G.

This means that for each column in the matrix, $\hat{J}$ is fixed and it corresponds to the column where the value was set to one previously. In addition, i varies from 1, .., N with $i \neq \hat{I}$.

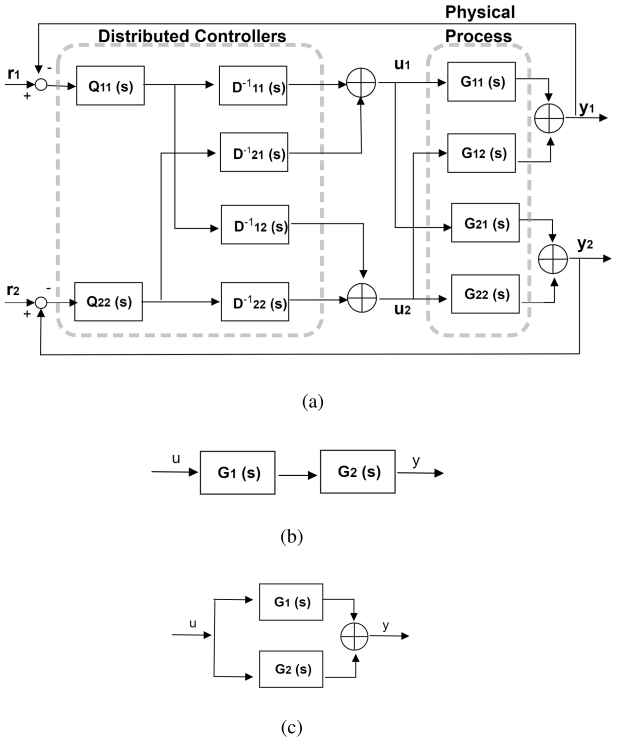

FIGURE 1. Decentralized models (a) via serial decomposition (b) or parallel decomposition (c).

Hence, each element $d_{i\hat{j}}$ is obtained from dividing the element $(i, \hat{J})$ in the $adjG(s)$ matrix between the value in the position $(\hat{I}, \hat{J})$ of the matrix $adjG(s)$.

We have to repeat this process for each column by fixing a new $\hat{J}$ to obtain the complete matrix $D(s)$ corresponding to one single model.

After we obtained the complete matrix $D(s)$, we repeat the whole process by selecting a different row $\hat{I}$ to obtain another model different from the previous one.

Hence, for a $n \times n$ process, there are n^n possible choices of $\hat{I}$ and $\hat{J}$. So, there are n^n possible $D(s)$ since it depends on the possible positions to place the $1s$ values when building matrix $D(s)$.

However, some of those choices can result in non-realizable systems. For example, if the adjudged matrix has a zero value in that entry. Thus, the configuration can be selected depending on the realizability.

Step 1.3: $Q(s)$ is a diagonal matrix built using Equation (7) and multiplying $G(s)\cdot D(s)$. Each matrix $D(s)$ gives, as a result, a different matrix $Q(s)$.

In Figure 1(a), we can the representation of the controllers architecture based on the defined matrix $Q(s)$ and $D^{-1}(s)$. Since we want to control the physical process defined by $G(s)$, the controllers will execute $Q(s)$ and $D^{-1}(s)$ due to Equation 7. Each entry of these matrices is the transfer function of one controller represented in the figure. Since Matrix Q(s) is a diagonal matrix, we have two controllers Q_{11} and Q_{22} that execute the transfer function in position (1,1) and (2,2) of matrix $Q(s)$. Then the output of these controllers Q_{ii} goes to controllers D_{ij}^{-1}. It corresponds with the product of matrices $Q(s).D^{-1}(s)$ since each element Q_{ii} multiplies row i in $D^{-1}(s)$ as follows.

$$\begin{bmatrix} q_{11} & 0 \\ 0 & g_{22} \end{bmatrix} \cdot \begin{bmatrix} d_{11}^{-1} & d_{12}^{-1} \\ d_{21}^{-1} & d_{22}^{-1} \end{bmatrix} = \begin{bmatrix} q_{11}d_{11}^{-1} & q_{11}d_{12}^{-1} \\ q_{22}d_{21}^{-1} & q_{22}d_{22}^{-1} \end{bmatrix}$$

In addition, considering Equation 1, we have that $G(s).u = Q(s).D^{-1}(s).u = y$. Hence, we have the following equation:

$$\begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} q_{11}d_{11}^{-1} & q_{11}d_{12}^{-1} \\ q_{22}d_{21}^{-1} & q_{22}d_{22}^{-1} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

The products of transfer functions are controllers in series which corresponds to a representation as in Figure 1 (b). In the previous equation q_{11} and d_{11}^{-1} are multiplied, so, in the Figure they are controllers in series.

On the other hand, the sum of the transfer function are parallel controllers which correspond to a representation as in Figure 1(c). For example, according to the previous equation we have the following result.

$$\begin{bmatrix} q_{11}d_{11}^{-1}u_1 + q_{11}d_{12}^{-1}u_2 \\ q_{22}d_{21}^{-1}u_1 + q_{22}d_{22}^{-1}u_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

For that reason, y_1 is expressed as the sum of two components that came from serial controllers.

As a result, Figure 1 provides the architecture of the designed system which is correlated with the physical models design (its transfer functions) and the network design, i.e. Q_{11} will communicate with D_{11}^{-1} and D_{12}^{-1}. But, it won' t communicate for example with Q_{22}.

Step 1.4: Due to realizability restrictions, it is possible to have matrices D with many elements equal to 0, which reduces the number of possible generated models. In this case, it is possible to generate other equivalent models using transfer function decomposition techniques.

Step 1.4.1 (Serial Decomposition): A transfer function $G(s)$ may be decomposed in transfer functions that multiply together as showed in Figure 1 (b). Hence, $G(s) = G_1(s).G_2(s)$. This decomposition is commutative and it is possible to generate combinations of the different factors to create the distributed transfer functions. This can be applied at the level of transfer functions as well as factoring the original transfer function in its poles and zeros representation as follows:

$$G(s) = k\prod_{i=1}^{N}\frac{s - z_i}{s - p_i} \tag{8}$$

where the denominator coefficients p_i are the poles, the numerator z_i are the zeros of the transfer function and k is the gain term. This mechanism allows generating different partitions of matrices $Q(s)$ and $D(s)$.

Step 1.4.2 (Parallel Decomposition): In this case, the transfer function $G(s)$ is decomposed into a sum of terms as showed in Figure 1 (c). Hence, $G(s) = G_1(s) + G_2(s)$. This can be done with a technique called partial fraction

decomposition that finds the residues and poles. The terms are as follows:

$$G(s) = k + \sum_{i=1}^{N} \frac{r_i}{s - p_i} \tag{9}$$

where the denominator coefficients p_i are called the poles of the transfer function, the numerator r_i is the residue of pole p_i and k is a constant. Hence, after applying this technique to a d_{ij} transfer function, we will obtain a family of d_{ij}^t functions that can be added to obtain the original d_{ij} function.

This mechanism allows generating a different distribution of compensators matrices $D(s)$.

To provide more misleading information to the attacker, one may add deceiving controllers that include more variability and mimic a real controller but they execute a transfer function that is compensated by the action of another controller.

Step 1.5: After calculating the sets of matrices $D(s)$ and $Q(s)$, it is possible to take each d_{ij} and q_{ij} entry to calculate its corresponding matrices A, B and C using the procedure to transform a transfer function into a state-space model.

The obtained matrices for each d_{ij}, will be called $A_{D_{ij}}$, $B_{D_{ij}}$ and $C_{D_{ij}}$. In a similar way, it is possible to take the q_{ii} values in $Q(s)$ and calculate its corresponding matrices A, B and C to obtain the matrices $A_{Q_{ij}}$, $B_{Q_{ij}}$ and $C_{Q_{ij}}$.

Step 2 (Network Design): In this section, we analyze how to design the network connectivity graph $\mathcal{G} = [V, E]$ for each of the physical models created in Step 1.

Step 2.1: The transfer functions in $Q(s)$ are controllers that take one input and send one output. Each of them will be executed in one node. For notation, if a node v_q executes the controller q_{ii} then we will call it $v_{q_{ii}}$.

The d_{ij}^{-1} and $d_{ij}^{-1\,t}$ elements take the output of the q_{jj} element to make their calculations and produce an output control signal. Each d_{ij}^{-1} will be executed in one node v_d and the notation will be $v_{d_{ij}}$ to express that the node v_d executes the transfer function d_{ij}^{-1}.

The network contains also a set of sensor nodes v_s and a set of actuator nodes v_a. If the sensor measures the variables of G_{ij}, then the notation will be v_{s_i}. In a similar way, v_{a_j} represents the actuator that applies the control input j.

Hence, the set of nodes V in the graph $\mathcal{G}$ contains the nodes v_q, v_d, v_s and v_a. In the system, there are also network devices, such as routers and switches. However, we are not explicitly including them in the design as we assume a traditional use of them.

Step 2.2: The set of edges E will be defined from the matrices $D(s)$ and $Q(s)$ according to the following four main rules: (1) $(v_{q_{ii}}, v_{d_{ij}}) \in E$; (2) $(v_{d_{ij}}, v_{a_i}) \in E$; (3) $(v_{d_{ij}^t}, v_{a_i}) \in E$; (4) $(v_{s_i}, v_{q_{ii}}) \in E$. An example can be observed in Figure 1(a) where according to the rule (1) the component q_{11} is connected to d_{11}^{-1} and d_{12}^{-1}. In addition, the output of q_{22} should be sent to d_{21}^{-1} and d_{22}^{-1}. Due to rule (2), the output of components d_{11}^{-1} and d_{21}^{-1} are combined to create the command u_1 that should be received by actuator a_1. In a similar manner, it is created the command for actuator a_2. Rule (3) is the equivalent to rule (2) when parallel decomposition is applied. In this particular case, it does not apply. Finally, rule (4) indicates that the sensor s_1 and s_2 measure the data that should be sent to components q_{11} and q_{22} respectively.

Step 2.3: To coordinate the system, there will be an orchestrator, physically located in the SDN controller. The orchestrator's responsibilities, such as the next model selection or the network reconfiguration, are detailed in Appendix A.

Step 3 (Switching Function Design): Next, it is required to design the switching function σ which indicates when to change the activated subsystem. In Appendix B we demonstrate that, from the physical point of view, it is possible to use an unrestricted switching signal, this means that there is no minimum switching time required since the proposed subsystem share a Common Quadratic Lyapunov function by design. Hence, this ensures the stability of the proposed switched linear system. However, in this type of system, the physical part is coupled with the cyber components and for this reason, the switching must be done considering the correct behavior of the cyber layer, for example, a switching time that allows the network devices to update correctly the routing tables.

V. EXPERIMENTAL WORK

A. TESTBED

We simulate a CPS using a simplified version of the Tennesse Eastman (TE) control challenge problem [47], already used in the related literature [48]. The system is described by the following matrix of transfer functions:

$$y = \begin{bmatrix} F4 \\ P \\ yA3 \\ VL \end{bmatrix} = G(s)u = \begin{bmatrix} g_{11} & 0 & 0 & g_{14} \\ g_{21} & 0 & g_{23} & 0 \\ 0 & g_{32} & 0 & 0 \\ 0 & 0 & 0 & g_{44} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{10}$$

where the monitored variables are the production rate (F4), the pressure (P), the amount of reactant A in the purge flow (yA3) and the liquid inventory (VL). The individual transfer functions are given below (the unit of s is seconds).

$$g_{11} = \frac{0.02833}{45s+1} \quad g_{21} = \frac{45(340s+1)}{9000s^2+615s+1}$$
$$g_{23} = \frac{-900s-11.25}{9000s^2+615s+1} \quad g_{32} = \frac{1.5}{600s+1}e^{-6s}$$
$$g_{14} = \frac{-3.4s}{360s^2+66s+1} \quad g_{44} = \frac{1}{60s+1}$$

B. DESIGN PROCEDURE

Given the system transfer function G(s) in Equation (10), we apply the proposed approach to obtain a resilient design to control the CPS.

Step 1.1 — Firstly, it is necessary to calculate $adjG(s)$ the adjudged matrix of G. In this particular case, we can observe that the output $yA3$ does only depend on variable u_3, i.e., row 3 and column 2 have all zeros except for the

element g_{32}. Hence, Steps 1.2 and 1.3 will give as a result the same function. To simplify the calculations, we will remove this row and column to obtain a G' matrix. We will add the component g_{32} again later in the process. The adjugate matrix for G' is as follows:

$$adjG' = \begin{bmatrix} g_{23}g_{44} & 0 & -g_{14}g_{23} \\ -g_{21}g_{44} & g_{11}g_{44} & g_{14}g_{21} \\ 0 & 0 & g_{11}g_{23} \end{bmatrix} \quad (11)$$

Step 1.2 — We calculate the matrix $D(s)$ column by column choosing a position for the unity value. Here, we will show the process just for the first column. Hence, we will design only the first controller, obtaining the controller Q_{11} and four compensators $D_{i1}, i = 1 \ldots 4$. This process is repeated with the other columns to obtain the other controllers.

To build the first column of matrix $D(s)$, we place the unit value in positions d_{11} or d_{21}. Notice that d_{31} equals 1 is a non realizable configuration due to $adjG'_{31} = 0$. We obtain two different physical models for Controller 1:

- Model (a): If we choose the option $d_{11} = 1$ then $d_{21} = adjG'_{21}/adjG'_{11} = -g_{21}/g_{23}$.
- Model (b): If we choose the option $d_{21} = 1$ then $d_{11} = adjG'_{11}/adjG'_{21} = -g_{23}/g_{21}$.

Step 1.3 — After this, we can calculate matrix $Q(s)$ for the calculated matrix $D(s)$ in the previous step. In this case, we are just doing one column of the complete matrix, i.e., we can calculate the corresponding controller Q_{11} by multiplying the first row of $G(s)$ and the first column of $D(s)$ to obtain Model (a) as $q_{11} = g_{11}$; and Model (b) as $q_{11} = -g_{11}g_{23}/g_{21}$.

Step 1.4.1 — In the previous steps, we obtained two models (a and b) for the distribution of controller Q_{11}. However, this does not generate enough models to create variability. In addition, the structure we want to build is formed for n controllers D_{ij} connected to each Q_{ii} controller. For this reason, we will apply series decomposition to generate more models and then parallel decomposition to generate four parallel controllers D_{ij}.

In a serial decomposition, we express the global transfer function $G(s)$ as a product of different factors that are executed in the different controllers obtained from the transfer functions $D(s)$ and $Q(s)$.

Table 1 summarizes the generated models using this technique. Model (c) has been generated starting from Model (a). According to equation 7, we have that $G(s) = Q(s) \cdot D^{-1}(s)$. At this point, we are creating both matrices and we have not applied the inverse operation to matrix $D(s)$ yet.

Hence, to create Model (c), we part from Model (a) and we move the factor $1/g_{23}$ from the transfer function d_{21} to the transfer function q_{11} as the inverse operation due to $G(s)$ will use the inverse of $D(s)$ in a future step. However, if we observe in Figure 1, when we change controller q_{11}, we also affect the result that goes to the transfer function d_{11} that is the reason why we have to multiply this controller for g_{23} also. As $G(s)$ uses the inverse of $D(s)$, we get that the changes of the entry d_{11} and q_{11} get compensated and the overall transfer function $G(s)$ does not change.

TABLE 1. Models generated with series decomposition.

Model	q_{11}	d_{11}	d_{21}
(a)	g_{11}	1	$-g_{21}/g_{23}$
(b)	$-g_{11}g_{23}/g_{21}$	$-g_{23}/g_{21}$	1
(c)	$-g_{11}g_{23}$	$-g_{23}$	g_{21}
(d)	$-g_{23}$	$-g_{23}/g_{11}$	g_{21}/g_{11}
(e)	$-g_{11}g^2_{23}/g_{21}$	$-g^2_{23}/g_{21}$	$1/g^1_{23}$

Similarly, we generate Model (d) from Model (b) using the factor g_{11}/g_{21} in q_{11} and moving its inverse to entries d_{11} and d_{21}.

More models can be generated if we apply this same technique but at the level of factors of the original transfer function. For example, we can obtain model (e) from model (b) in the following manner. The transfer function g_{23} can be expressed using the poles and zero representation as follows.

$$g_{23} = \frac{-900s - 11.25}{9000s^2 + 615s + 1} = -0.1\frac{s + 0.0125}{(s + 0.0667)(s + 0.0017)}$$

The poles and zeros representation can be calculated using *tf2zp* function in Matlab. Hence, it is possible to rewrite $g_{23} = g^1_{23} \cdot g^2_{23}$ where g^1_{23} and g^2_{23} are any combination of the previous factors, for example, one of them may be as follows.

$$g^1_{23} = -0.1\frac{s + 0.0125}{(s + 0.0667)} \qquad g^2_{23} = \frac{1}{(s + 0.0017)}$$

In this way, it is possible to move factors from the transfer function q_{11} to d_{11} and d_{21} by applying the inverse operation as in the previous examples. In this way, it is possible to obtain even further models as we obtained (e).

Step 1.4.2 — After the previous step, we have many different models for q_{11}. However, we have just two d_{ij} because d_{31} and d_{41} are zero in $D(s)$. To improve this, we can apply partial fraction decomposition. We will show the procedure for Model (c). The component d_{11} can be separated in the following transfer functions:

$$d_{11} = \frac{900s + 11.25}{9000s^2 + 615s + 1} = \frac{0.0833}{s + 0.0667} + \frac{0.0167}{s + 0.0017}$$

Hence, we can divide d_{11} in the addition of two transfer functions:

$$d^1_{11} = \frac{0.0833}{s + 0.0667} \text{ and } d^2_{11} = \frac{0.0167}{s + 0.0017}$$

Similarly, we can transform d_{21} using the partial fraction decomposition as follows.

$$d^1_{21} = \frac{1.6667}{s + 0.0667} \text{ and } d^2_{21} = \frac{0.0333}{s + 0.0017}$$

With this procedure, we found the four compensators $d_{i1}, i \in 1..4$. The partial fraction decomposition can be found using the *residue* function in Matlab using the transfer functions.

Step 1.5 — Matrices A, B and C, for each transfer function, are not included in this article due to space limits but they can be easily obtained using Matlab functions *ss*, *c2d* and *ssdata* from the transfer function.

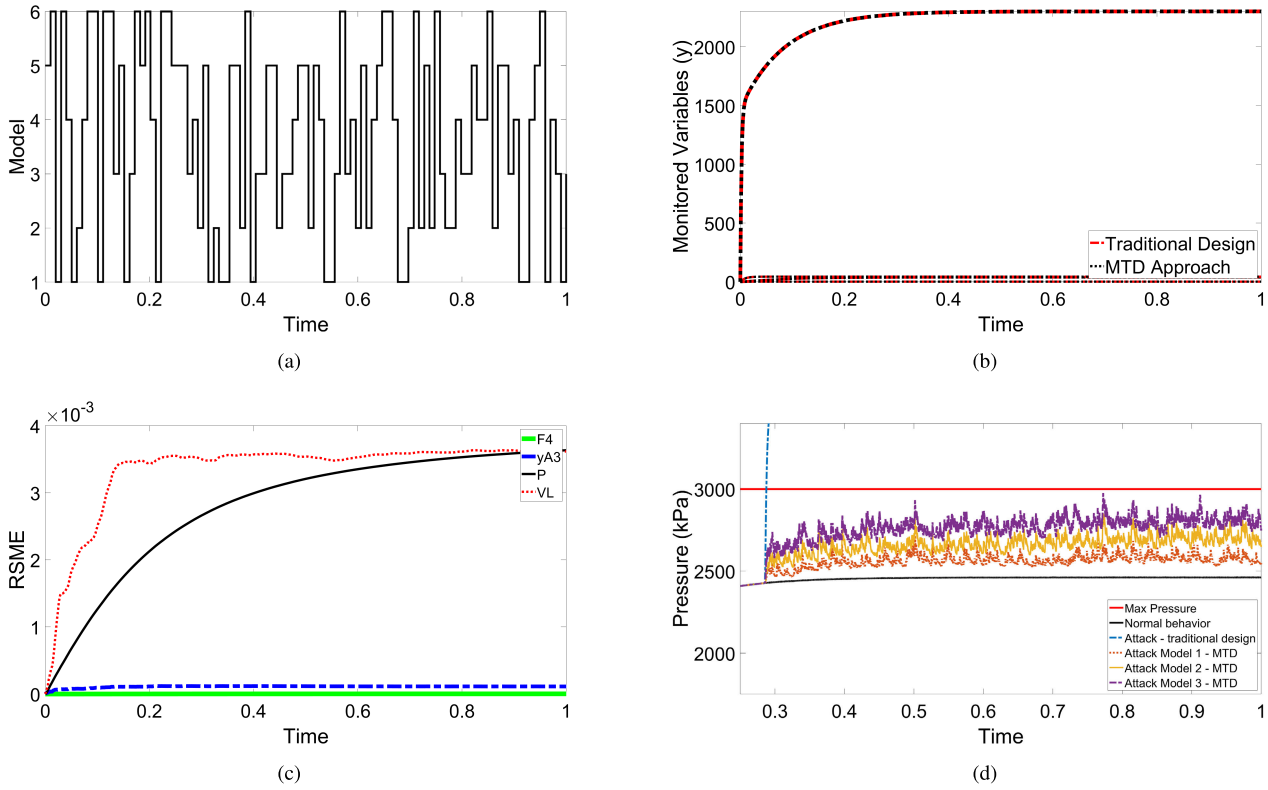

FIGURE 2. Experimental results. Time (x-axis) is normalized between 0.0 and 1.0, representing the temporal percentage of multiple experimental runs. (a) MTD switching signal over time. (b) Evolution of the system states for the traditional design without MTD and with our MTD approach. (c) Optimality loss. Root Mean Square Error of the MTD approach with respect to the traditional design. (d) Pressure evolution under attack with and without the MTD approach.

Step 2.1 and 2.2 — The control theory diagram of the obtained system is similar to the one showed in Figure 1 (a) where there are four Q_{jj} boxes that execute the transfer function in the position (j, j) of the matrix $Q(s)$ and each one is connected to four D_{ij} that execute the transfer function in the position (i, j) of the matrix $D^{-1}(s)$.

Step 2.3 — This point describes the controller dynamic behavior.

Step 3 — Using the Matlab function *dlyap* it was verified that condition (12) is met and in consequence, the system will be stable under unrestricted switching.

C. RESULTS

To validate the approach, we implemented a numeric simulation with Simulink. From all the possible derived models, we choose V of them in an aleatory way to create a reduced proof of concept of the resilient CPS and analyze how the system reacts to adversaries with different capabilities. In this case, V is set up to six. However, in Section V-D, we analyze the possible model generation for a process $n \times n$. The feedback loop was implemented using a Linear Quadratic Gaussian (LQG) approach.

Results are shown in Figure 2. All the plots depicted in Figure 2 assume that time (cf. x-axis) is normalized between 0.0 and 1.0, representing the temporal percentage of multiple experimental runs. Figure 2(a) shows the MTD switching signal that selects the model to execute over time. The switching signal is configured at a frequency of 1 over 10. To simplify the simulation, the switching time was set up periodic. However, this is not necessary and it is possible to use non-periodic signals. Figure 2(b) shows the evolution of the system states in normal behavior, i.e., without malicious actions, applying the proposed MTD approach and the traditional design without the MTD approach. It is possible to verify that the system remains stable and equal to the traditional system design although the model switching. In Figure 2(c), we present the Root Mean Square Error (RMSE) to analyze the optimality loss due to the new design. We can observe that the error between both signals, the one with the traditional system design and the proposed MTD approach, is in the order of 10^{-3}.

The actuators are valves that should operate in the range 0-100% which corresponds to the saturation limits. The process has to operate under certain safety constraints. One of them is that the reactor pressure should not exceed 3000 kPa [49].

The adversary aims at damaging the physical process. Hence, his objective will be to make the process pass the pressure limit to damage the system pipes. The pressure is monitored by the output P and in Equation (10), it is possible to see that it depends on control inputs u_1 and u_3 since

TABLE 2. Attack scenarios and pressure increase.

Reference	Known Models	Max Pressure Increase
M1	15%	10.41%
M2	30%	15.93%
M3	50%	20.98%

$P = g_{21}.u_1 + g_{23}.u_3$. In addition, g_{21} has a positive sign. So, if we increase u_1, we will increase the pressure. On the contrary, g_{23} has a negative sign, so we need to decrease u_3 value to increase the pressure.

These control inputs are managed by the controllers Q_{11} and D_{1j} for u_1, and Q_{33} and D_{3j} for u_3 with $j \in 1..4$. The most efficient and powerful adversary is the one capable of compromising, in the case of u_1 the outputs from the D_{1j} controllers and the input of Q_{11} and analogously for u_3. This adversary is the one that we implemented to test the approach since it is the worst-case. In addition, we defined adversaries with different capabilities in terms of the number of models that they are capable of learning for those compromised controllers and the saturation level of the valves. We consider adversaries that are capable of learning 15% of the models (Model 1), 30% (Model 2) and 50% (Model 3). Also, for the saturation level, we consider u_1 and u_3 completely saturated at 100% and 0% respectively. As a reference point, the saturation level for the valves at the normal case and in stability conditions are 60.95% for u_1 and 25.02% for u_3.

Table 2 shows the maximum pressure increase for the worst-case adversary with respect to the normal case. In the case of an attack without a resilience approach, the system's pressure increases 21.95% reaching the maximum possible and damaging the pipes.

Figure 2(d) shows the pressure threshold, the system pressure in normal conditions and under attack considering a traditional design and a MTD design facing adversaries Model 1, 2 and 3. The attack starts when the system is already stable. It is possible to observe that the traditional design is not resilient and the adversary is able to make the system move to the unsafe condition passing the threshold. In the system designed with the MTD approach, the adversaries are not able to make the system exceed the maximum pressure. The process signal presents little oscillations due to the correct models that compensate the actions of the adversary that tries to move the process out of stability.

D. EVALUATION OF RESULTS

1) ATTACK SURFACE AND DEFENSES

The attack surface of a system can be seen as the subset of resources that an attacker can use to attack the system. This includes the entry and exit points of the system, its channels and any untrusted data items exchanged with the system.

According to the adversary defined in Section II and the attack surface defined in [50], the relevant resources that we have to protect are the system measurements (entry points), the command inputs (exit points) that are exploited using the data network packet payloads (channel) and that are generated exploiting the knowledge that the adversary has about the controller model (untrusted data items).

The cyber-physical attacks start with a reconnaissance phase to gather intelligence about the system. This requires time and effort for the attacker.

Remark 1: The resilience approach attempts to render the attacker's intelligence invalid by switching the used physical model and remapping the network addresses. Our strategy protects the resources by continuously shifting the attack surface using diversity defenses at two levels: node and network. At the physical level, the approach converts a centralized Linear Time Invariant system into an equivalent distributed Linear Time Variant system using switched rules, i.e., from a unique controller represented as in Equation (1) by $G(s)$, we obtain a distributed design determined by matrices $Q(s)$ and $D(s)$ which provides $n(n+1)$ controllers. In addition, these $n(n+1)$ controllers switch the models over time and in consequence, they modify the logic for creating the data payloads of the packet since the commands respond to a new distributed way of calculating them. At the network level, the devices change the endpoint information to deceive the adversary.

Remark 2: The system configuration switching should be done with enough regularity to make any information collected with reconnaissance purposes expire quickly. It aims at developing a mechanism that continually and unpredictably changes the parameters of the system to increase the cost of attacking, limit the exposure of vulnerable components and deceive the opponent.

2) ATTACK STRATEGIES AND SUCCESS ANALYSIS

The effectiveness of a moving target defense depends on the attacker's capabilities and strategy.

Remark 3: In cyber-physical exploits the adversary payload contains a set of instructions that manipulate the process and the choice of instructions depends on the specific impact the attacker wants to have on the process.

Hence, in this section, we consider different strategies an attacker may employ against the system defenses. According to [51], the phases to achieve a cyber-physical attack are as follows. First, it is the *Access* phase which is the traditional hacking that gives the adversary an entry point to be inside the system. This part of the attack is not relevant for our analysis since it is related to classical cybersecurity problems. Then it is the *Discovery* phase where the adversary tries to learn how the system was designed and built. The next phase is the *Control* where the adversary tries to discover the dynamic behavior of the process that can be described by the transfer function or state-space model which are related by cause and effect relationships of the process. Finally, is the *Damage* phase where the attacker performs the attack itself.

Next, we discuss two adversary types with different knowledge capabilities and their strategies to overcome the attack phases described previously. One of the adversaries has no knowledge and performs a *brute force* attack. The other has

detailed knowledge of the system and performs an *efficient targeted* attack.

3) DISCOVERY

The brute force adversary starts with access to a CPS network but he has no knowledge about the system. During the discovery, the adversary collects information about the system to learn about its structure, how it works and how it was built. It is necessary to learn which are the components and how they are interrelated from analyzing network traffic. In this stage, the adversary faces the first and simplest barrier which is the network MTD that modifies the device's IP addresses. If there are K devices and each of them has a range R of available IP addresses, the adversary has to recreate the network topology without any knowledge about how many real devices there are. The adversary needs to learn which type of sensors and actuators are involved in the process, and guess which is the function of the physical process, how it works and which may be the safety conditions to be exploited.

The efficient targeting adversary has much more detailed knowledge about the physical process which is more difficult and time consuming to obtain. This attacker may be an ex-employee or someone who has access to the system management documentation. First, he studies general information about the physical part such as chemistry, kinetics, thermodynamics, etc. This can be done by consulting open literature as well as proprietary information of process design companies. He may have typical company internal documents about system design, such as the ones described in [51]. Piping and Instrumentation Diagrams which contains the system layout and physical structure, One-Line Diagrams which often contain information on safety conditions, Cause and Effect Diagrams with the behavior of the system, Cable Diagrams with the physical network topology, Instrument input/output Lists contain a list of instruments which serve as input or output of the control system, among others. Hence, this adversary has precise knowledge about how the system carries out its functions, how it was built and the conditions that can put the system in danger.

For the efficient targeting adversary, the network MTD should not be a major problem since the adversary knows that exist K devices and their functions. Hence, performing some network analysis, the adversary may guess it.

4) CONTROL

In the CPS resilient design, there are $n(n+1)$ controllers and n^n possible physical models available for the factorization in the matrices $D(s)$ and $Q(s)$. However, some of these configurations will not be realizable and the number of available factorization is given by:

$$\prod_{j=1}^{p}(\#TF \times \#DS \times \#DP)$$

where p is the number of actuators, i.e., the number of columns in the matrix $G(s)$, $\#TF$ corresponds to the number of transfer functions different from zero in the column j of the adjudged matrix, $\#DS$ corresponds to the number of possible series decomposition of a transfer function to generate two new transfer function which is C_w^2 combination of two taken from w, where w is the transfer function polynomial grade. In addition, $\#DP$ corresponds to the number of parallel decompositions which are $\sum_{j=1}^{p} C_w^j$ where p is the number of control signals.

Remark 4: To be successful, i.e., to go unnoticed during the attack, the adversary has to learn the models of the other cascade dependent controllers. For example, in figure 1, if the adversary manages to learn the model of controller Q_{11} and deceive it. The value that D_{11} and D_{21} receive will not be correlated with what they expect and it is possible to know that something is not working properly in the system. In a similar way, if the attacker learns the model of D_{11} and manages to insert malicious messages, those commands will be executed by the actuators that affect G_{11} and G_{21}, which will modify the measures y_1 and y_2. Hence, Q_{11} and Q_{22} will receive values that are not the expected values.

For this reason, it is not enough to learn just one model, in every switching period the adversary has to gain a position in the required network links to learn the models of all the correlated close-loops to go unnoticed. Hence, in this phase, both adversaries have to do the same work. However, the efficient targeted adversary can perform a smarter strategy. Since he knows which safety condition he wants to exploit and which are the controllers involved in controlling that variable, he needs to compromise those involved controllers. However, the cascade effect in correlated close-loops will force him to consider also the other controllers too and as a result, his work will not be easier than the brute force adversary.

On the contrary, the brute force adversary has no knowledge about the safety condition he may exploit. So, he has to learn all the controllers' models and start doing small probes. In this way, by injecting smart disturbances he needs to understand how all the components work together and the cause-effect of the system variables to create a strategy to damage the system. If the switching time is big enough, such a learning process may be practical. Estimating the time required for an attacker to gather sufficient knowledge during the control phase is critical to assess the attacker's ability to successfully compromise the system and allow us to disrupt the attacker's reconnaissance effort. In this way, it is possible to set up the switching time to avoid the learning.

Remark 5: The adversary, in the most efficient scenario, has to (1) rebuild the network topology, (2) collect network traffic and (3) use this data to learn the model, for example, using machine learning. The time required for (1) can be depreciated for the efficient target adversary. However, Tasks (2) and (3) involve tasks that require in the order of several minutes to be performed.

Remark 6: Learning one model for just one controller involves learning many independent variables, i.e., the system parameters mentioned in Section II.1, matrices A, B, C, Q and R. Hence, the complexity of learning one model increases

significantly with the complexity of the physical process, i.e., if the system has more sensors and actuators.

Remark 7: The time required for a model switching can be in the order of the seconds to leave enough time to converge the network devices in charge of the packets forwarding. Hence, this can make the task of the adversary hard to achieve.

Each time the attacker learns a model it gains some knowledge that can be used when the same model is executed again. In this case, the adversary has to guess the switching signal or he needs to gather data from each switching period to test if the current model fits with one of the previously learned models. Hence, the models already learned in the previous periods reduce the required effort for the adversary. However, the time required to learn each new model is not reduced because of the knowledge of previous models.

5) DAMAGE

Even if the adversary learns the models and injects malicious packets during a switching period, i.e., the adversary turns that period into an unstable one, the system can still ensure stability as demonstrated in [52]. To be successful, the adversary has to compromise more than 50% of the physical models. If the adversary learns less than 50%, the stable model is activated sufficiently long (i.e., it is possible to absorb the state divergence made by unstable modes).

E. DISCUSSION

The example presented in Section V.1 is a simplified version of a whole chemical process. The complete TE system has 50 states, 41 measured variables and 12 control inputs. Hence, it is possible to generate 12^{41} models (i.e., approximately 2^{147} models). If we switch the model every 30 seconds, the adversary will need 1.6×10^{38} years to learn all models. Another well-known testbed such as the Secure Water Treatment (SWaT) system has 51 devices including sensors and actuators. The Vynil Acetate Monomer (VAM) Process has 246 states, 43 measured variables and 26 control inputs. In addition, a real industrial system may have even more devices. Hence, when applying this technique to bigger processes, it is possible to derive more models and get quite robust designs.

We have provided a concrete case showing how to apply the MTD approach where all the generated models are equivalents to the original one. The fact of building them through equivalences makes it easier to ensure the stability of the process but it may limit the number of models that we can generate to apply the approach. However, this mechanism can go further since the equivalence of the models is not a strict requirement. Actually, it is possible to switch different stable or unstable subsystems and ensured the stability of the global system if the switching signal is designed properly.

The control theory community has mathematically proved different switching stabilization methods for both stable and unstable subsystems [35], [39]. For example, in [52], the authors prove that if the total activating period of unstable modes is small enough compared with that of stable modes, the stability of switched linear systems is guaranteed. In addition, as showed in [53] and [54], it is also possible to design the system to switch all unstable subsystems and get as a result a stable system. In consequence, the model generation can be much wider than the one presented in this article but to do so it is required to determine in a practical manner how to build the models starting from the initial transfer function and how to design the proper switching function to control the physical process while guarantying the global stability of the system.

VI. CONCLUSION

We have presented a moving target defense (MTD) approach to design resilient cyber-physical systems (CPS). The approach has been modeled using switching linear control. A series of decentralized controllers periodically modify the underlying physical and network configuration models of the CPS, satisfying self-healing properties. The approach has been validated using the Tennesse Eastman problem. We have simulated and validated the approach. The obtained results are very promising. The resulting design is capable of absorbing and recovering from attacks while guaranteeing the stability of the physical process. At the same time, as we have shown in our evaluation results, the approach makes more complex the tasks that the adversary should do to be successful at performing new attacks.

Further work will focus on quantifying new adversarial capabilities, both in terms of temporal and spatial constraints. Another future direction involves developing further mechanisms for model generation that create randomness in the system from the adversary point of view, while still being able to ensure the stability of the physical process. Finally, it is necessary to build an approach to compare systems' design to evaluate the resilience improvement that different proposal can achieve.

APPENDIX A
APPENDICES

A. ORCHESTRATOR RESPONSIBILITIES

The responsibilities of the orchestrator are described as follows:

1) **Choosing a key for the model selection**. There are $\mathcal{I} = \{1, \ldots, N\}$ possible subsystems to activate and the orchestrator chooses randomly a key K_1 which will be used to select the next model to activate using a hash function as follows $hash(K_1, j)\ mod\ N$ where j is the switching interval. The common sharing of K_1, j and N allows each device to compute the next active model in a distributed manner. The key is renewed periodically using one of the existing approaches for key generation and distribution such as [55].
2) **Coordinating the network configuration transformation.** Each component will change its network configuration in each switching period of the physical model. To do this, each device gets a real IP address (RIPA) and a virtual IP address (VIPA).

The RIPA is used for management purposes making the network configuration transformation transparent to administrators. The VIPA is used to communicate the data packets of the CPS, i.e., the hosts communicate with another host using their VIPAs. In addition, VIPAs change periodically and synchronously in a distributed fashion over time. In every transformation interval, the hosts will be associated with a unique VIPA.

The VIPA transformation is managed by the SDN devices by selecting an address from the unused address space. Each host will be allocated an IP address ranges to choose the VIPAs and they are selected using a hash function from the designated ranges. Since the VIPAs are chosen from the assigned network sub-nets, there is no need to do a routing update advertisement for internal routers. In addition, SDN devices will forward packets from old connections until the session is terminated or expired.

Each SDN device is responsible for the management of the hosts in one or more sub-nets. The VIPAs selection is done in a similar way to the physical model selection. It uses a hash function and a secret random key to guarantee unpredictability. If there are p available VIPAs for a host, then the SDN device can compute the index of the VIPA for the switching interval j as $hash(K_2, j)\ mod\ p$. The SDN controller is responsible for the management of the SDN devices and the key K_2 distribution.

3) **Coordinating the transformation time**. The orchestrator has to choose and coordinate the switching in a master-slave mode. It requires a distributed timing synchronization that ensures the achievement and maintenance of a common time for all the nodes of the network. Many proposals have already work in solving this type of issue [56].

B. SWITCHING FUNCTION DESIGN

The stability of switched systems depends not only on the dynamics of each subsystem but also on the properties of the switching signals. For example, even when all the subsystems are stable, the switched system may have divergent trajectories for certain switching signals. In addition, it may be possible to switch between unstable subsystems to make the resulting switched system stable [44]. If one stays at stable subsystems long enough and switches less frequently, one may trade off the energy increase caused by the switching itself or the unstable modes, and maintain the stability of the system.

The switching function may depend on different parameters, such as the time instant k, the current state x_k, the output y_k or the previous active mode $\sigma(\tau)$ for $\tau < k$. However, during an attack, the state or the system output that a controller gets, may not be accurate with respect to the real state in the physical process. For this reason, it is desired that the switching function depends only on the time instant k. There are many approaches to analyze the stability of a system. In particular, Lyapunov stability theory [57] is based on the idea that at a stable equilibrium, the energy of the system has a local minimum, whereas at an unstable equilibrium, it is at a maximum. It analyzes the behavior of the system in the following form $x_{k+1} = \mathcal{A}x_k$, where $\mathcal{A}$ corresponds to the matrix of the system in an open-loop form executing the defined close-loop inside.

In addition, it is defined a scalar function $V(x)$ which is a Lyapunov function using a quadratic form $V(x) = x^T Px$, where P is a symmetric matrix, positive defined, i.e., all the eigenvalues of P are positive.

The Lyapunov Theorem states that a linear time-invariant discrete-time system $x_{k+1} = \mathcal{A}x_k$ is asymptotically stable if and only if for any positive definite matrix Q, such that $Q = Q^T > 0$ there exists a unique positive definite solution P to the discrete Lyapunov equation:

$$\mathcal{A}^T P\mathcal{A} - P = -Q < 0 \tag{12}$$

If this condition meets, the matrix $\mathcal{A}$ is asymptotically stable, i.e., all its eigenvalues have a negative real parts.

This theorem applies when we have a unique control model. However, in this case we have a switched linear system that is composed of a piecewise signal that we want to make stable although the model switching. For this reason, it is necessary to apply a variation of this theorem and find a Common Quadratic Lyapunov function for all the subsystems. In this case, we look for a positive definite symmetric matrix P such that

$$\mathcal{A}_i^T P\mathcal{A}_i - P = -Q_i < 0, i \in \mathcal{I} \tag{13}$$

This condition means that it is required to find one matrix P capable of fulfilling this property for all the subsystems. If this condition is met, the system will be asymptotically stable under arbitrary switching, i.e., there is no restriction on the switching signal [39].

The open-loop transfer function for the approach is determined by the equation $Q_i(s).D_i^{-1}(s)$. These matrices have been built according to three decomposition techniques. The first one is separate $G(s)$ in distributed controllers, this transformation is given in Equation (7) where it is possible to verify that the obtained matrices $Q_i(s)$ and $D_i(s)$ are equal to the original transfer function $G(s)$. The other applied transformation is the serial decomposition given by Equation (8) where it is also possible to verify that the product of the obtained components respect also the original transfer function. Finally, the same occurs with the decomposition of parallel serial function whose equation is Equation (9). For this reason, it is possible to conclude that $Q_i(s).D_i^{-1}(s) = G(s)$, $\forall i \in \mathcal{I}$. This means that the open-loop transfer function of the approach depends only of the original transfer function $G(s)$ which we know that is stable due to the initial assumptions made in Section II. Hence, exists a matrix P solution for condition 12. Finally, since all the subsystems are equivalents to $G(s)$, the same solution holds for condition 13 and the switched system is stable under arbitrary switching.

As a conclusion, and from the physical point of view, there are no restrictions for the switching signal that compromise the stability of the system. However, in this type of system, the physical part is coupled with the cyber components and for this reason, the switching must be done considering the correct behavior of the cyber layer.

REFERENCES

[1] X. Ge, F. Yang, and Q.-L. Han, "Distributed networked control systems: A brief overview," *Inf. Sci.*, vol. 380, pp. 117–131, Feb. 2017.

[2] V. L. Do, L. Fillatre, I. Nikiforov, and P. Willett, "Security of SCADA systems against cyber–physical attacks," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 32, no. 5, pp. 28–45, May 2017.

[3] H. S. Sánchez, D. Rotondo, T. Escobet, V. Puig, and J. Quevedo, "Bibliographical review on cyber attacks from a control oriented perspective," *Annu. Rev. Control*, vol. 48, pp. 103–128, 2019.

[4] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Proc. 1st Int. Conf. High Confidence Netw. Syst. (HiCoNS)*, Beijing, China, 2012, p. 55.

[5] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, Jan. 2015.

[6] *DHSR Lexicon*, Risk Steering Committee, US Dept. Homeland Secur., Washington, DC, USA, Sep. 2008.

[7] D. K. Mishra, M. J. Ghadi, A. Azizivahed, L. Li, and J. Zhang, "A review on resilience studies in active distribution systems," *Renew. Sustain. Energy Rev.*, vol. 135, Jan. 2021, Art. no. 110201.

[8] S. Weerakkody, O. Ozel, Y. Mo, and B. Sinopoli, "Resilient control in cyber-physical systems: Countering uncertainty, constraints, and adversarial behavior," *Found. Trends Syst. Control*, vol. 7, nos. 1–2, pp. 1–252, 2020.

[9] D. A. S. Estay, R. Sahay, M. B. Barfod, and C. D. Jensen, "A systematic review of cyber-resilience assessment frameworks," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101996.

[10] A. Kanellopoulos and K. Vamvoudakis, "A moving target defense control framework for cyber-physical systems," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1029–1043, Mar. 2020.

[11] J. Giraldo, A. Cardenas, and R. G. Sanfelice, "A moving target defense to detect stealthy attacks in cyber-physical systems," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2019, pp. 391–396.

[12] S. Weerakkody and B. Sinopoli, "A moving target approach for identifying malicious sensors in control systems," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Monticello, IL, USA, Sep. 2016, pp. 1149–1156.

[13] J. Rubio-Hernan, L. De Cicco, and J. Garcia-Alfaro, "On the use of watermark-based schemes to detect cyber-physical attacks," *EURASIP J. Inf. Secur.*, vol. 2017, no. 1, pp. 1–25, Jun. 2017.

[14] K. Paridari, N. O'Mahony, A. El-Din Mady, R. Chabukswar, M. Boubekeur, and H. Sandberg, "A framework for attack-resilient industrial control systems: Attack detection and controller reconfiguration," *Proc. IEEE*, vol. 106, no. 1, pp. 113–128, Jan. 2018.

[15] S. Hasan, A. Dubey, G. Karsai, and X. Koutsoukos, "A game-theoretic approach for power systems defense against dynamic cyber-attacks," *Int. J. Electr. Power Energy Syst.*, vol. 115, Feb. 2020, Art. no. 105432.

[16] L. Huang and Q. Zhu, "A dynamic games approach to proactive defense strategies against advanced persistent threats in cyber-physical systems," *Comput. Secur.*, vol. 89, Feb. 2020, Art. no. 101660.

[17] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving target defense techniques: A survey," *Secur. Commun. Netw.*, vol. 2018, Jul. 2018, Art. no. 3759626.

[18] K. Ogata, *Modern Control Engineering*, 4th ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[19] P. J. Campo and M. Morari, "Achievable closed-loop properties of systems under decentralized control: Conditions involving the steady-state gain," *IEEE Trans. Autom. Control*, vol. 39, no. 5, pp. 932–943, May 1994.

[20] J. Rubio-Hernan, R. Sahay, L. De Cicco, and J. Garcia-Alfaro, "Cyber-physical architecture assisted by programmable networking," *Internet Technol. Lett.*, vol. 1, no. 4, p. e44, Jul. 2018.

[21] M. Segovia, A. Cavalli, N. Cuppens, J. Rubio-Hernan, and J. Garcia-Alfaro, "Reflective attenuation of cyber-physical attacks," in *Proc. 5th Workshop Secur. Ind. Control Syst. Cyber-Phys. Syst. (CyberICPS), 24th Eur. Symp. Res. Comput. Secur. (ESORICS)*, vol. 11980. Berlin, Germany: Springer, Feb. 2020, pp. 19–34.

[22] J. Zheng and A. S. Namin, "A survey on the moving target defense strategies: An architectural perspective," *J. Comput. Sci. Technol.*, vol. 34, no. 1, pp. 207–233, Jan. 2019.

[23] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization," *Comput. Netw.*, vol. 51, no. 12, pp. 3471–3490, Aug. 2007.

[24] E. Al-Shaer, Q. Duan, and J. Jafarian, "Random host mutation for moving target defense," in *Security and Privacy in Communication Networks*, (A. D. Keromytis and R. Di Pietro, Eds. Berlin, Germany: Springer, 2013, pp. 310–327.

[25] D. C. MacFarland and C. A. Shue, "The SDN shuffle: creating a moving-target defense using host-based software-defined networking," in *Proc. 2nd ACM Workshop Moving Target Defense*, New York, NY, USA, 2015, pp. 37–41.

[26] S. Dolev and S. T. David, "SDN-based private interconnection," in *Proc. IEEE 13th Int. Symp. Netw. Comput. Appl.*, Aug. 2014, pp. 129–136.

[27] A. Aseeri, N. Netjinda, and R. Hewett, "Alleviating eavesdropping attacks in software-defined networking data plane," in *Proc. 12th Annu. Conf. Cyber Inf. Secur. Res. (CISRC)*, New York, NY, USA, 2017, p. 1.

[28] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient random route mutation considering flow and network constraints," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2013, pp. 260–268.

[29] D. Ma, C. Lei, L. Wang, H. Zhang, Z. Xu, and M. Li, "A self-adaptive hopping approach of moving target defense to thwart scanning attacks," in *Information and Communications Security*, K.-Y. Lam, C.-H. Chi, and S. Qing, Eds. Cham, Switzerland: Springer, 2016, pp. 39–53.

[30] P. E. Veríssimo, N. Neves, and M. Correia, "Intrusion-tolerant architectures: Concepts and design," in *Architecting Dependable Systems*, R. de Lemos, C. Gacek, and A. Romanovsky, Eds. Berlin, Germany: Springer, 2003, pp. 3–36.

[31] S. Forrest, A. Somayaji, and D. H. Ackley, "Building diverse computer systems," in *Proc. 6th Workshop Hot Topics Operating Syst.*, Washington, DC, USA, May 1997, pp. 67–72.

[32] A. R. Cavalli, A. M. Ortiz, G. Ouffoué, C. A. Sanchez, and F. Zaïdi, "Design of a secure shield for Internet and Web-based services using software reflection," in *Web Services–ICWS*, H. Jin, Q. Wang, and L.-J. Zhang, Eds. Cham, Switzerland: Springer, 2018, pp. 472–486.

[33] P. Griffioen, S. Weerakkody, and B. Sinopoli, "A moving target defense for securing cyber-physical systems," *IEEE Trans. Autom. Control*, early access, Jun. 29, 2020, doi: 10.1109/TAC.2020.3005686.

[34] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Syst.*, vol. 19, no. 5, pp. 59–70, Oct. 1999.

[35] H. Yang, B. Jiang, and V. Cocquempot, *Stabilization of Switched Nonlinear Systems With Unstable Modes* (Studies in Systems, Decision and Control), vol. 9. Cham, Switzerland: Springer, 2014.

[36] D. Liberzon, *Switching in Systems and Control*. Boston, MA, USA: Birkhäuser, 2012.

[37] J. P. Hespanha and A. S. Morse, "Switching between stabilizing controllers," *Automatica*, vol. 38, no. 11, pp. 1905–1917, Nov. 2002.

[38] S. Pettersson, "Synthesis of switched linear systems," in *Proc. 42nd IEEE Int. Conf. Decis. Control*, vol. 5. Maui, HI, USA, Dec. 2003, pp. 5283–5288.

[39] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: A survey of recent results," *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 308–322, Feb. 2009.

[40] J. P. Hespanha and A. S. Morse, "Stability of switched systems with average dwell-time," in *Proc. 38th IEEE Conf. Decis. Control*, vol. 3. Phoenix, AZ, USA, Dec. 1999, pp. 2655–2660.

[41] G. Zhai, B. Hu, K. Yasuda, and A. N. Michel, "Qualitative analysis of discrete-time switched systems," in *Proc. Amer. Control Conf.*, vol. 3, May 2002, pp. 1880–1885.

[42] A. N. Michel, "Recent trends in the stability analysis of hybrid dynamical systems," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 46, no. 1, pp. 120–134, Jan. 1999.

[43] H. Ye, A. N. Michel, and L. Hou, "Stability theory for hybrid dynamical systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 461–474, Apr. 1998.

[44] R. A. Decarlo, M. S. Branicky, S. Pettersson, and B. Lennartson, ''Perspectives and results on the stability and stabilizability of hybrid systems,'' *Proc. IEEE*, vol. 88, no. 7, pp. 1069–1082, Jul. 2000.
[45] L. Liu, S. Tian, D. Xue, T. Zhang, Y. Chen, and S. Zhang, ''A review of industrial MIMO decoupling control,'' *Int. J. Control, Autom. Syst.*, vol. 17, no. 5, pp. 1246–1254, May 2019.
[46] Q. Wang, *Decoupling Control* (Lecture Notes in Control and Information Sciences), no. 285. New York, NY, USA: Springer, 2002.
[47] N. Lawrence Ricker, ''Model predictive control of a continuous, nonlinear, two-phase reactor,'' *J. Process Control*, vol. 3, no. 2, pp. 109–123, May 1993.
[48] R. Chabukswar, B. Sinópoli, G. Karsai, A. Giani, H. Neema, and A. Davis, ''Simulation of network attacks on SCADA systems,'' in *Proc. 1st Workshop Secure Control Syst.*, Apr. 2010, pp. 1–8.
[49] N. Lawrence Ricker, ''Decentralized control of the tennessee eastman challenge process,'' *J. Process Control*, vol. 6, no. 4, pp. 205–221, Aug. 1996.
[50] S. Jajodia, Ed., *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. *Advances in Information Security*, no. 54. New York, NY, USA: Springer, 2011.
[51] M. Krotofil and J. Larsen, ''Rocking the pocket book hacking chemical plants for competition and extortion,'' in *Proc. DefCon Conf. (DEFCON)*, 2015, p. 52.
[52] G. Zhai, B. Hu, K. Yasuda, and A. N. Michel, ''Stability analysis of switched systems with stable and unstable subsystems: An average dwell time approach,'' in *Proc. Amer. Control Conf. (ACC)*, vol. 1. Chicago, IL, USA, Jun. 2000, pp. 200–204.
[53] W. Xiang and J. Xiao, ''Stabilization of switched continuous-time systems with all modes unstable via dwell time switching,'' *Automatica*, vol. 50, no. 3, pp. 940–945, Mar. 2014.
[54] X. Mao, H. Zhu, W. Chen, and H. Zhang, ''New results on stability of switched continuous-time systems with all subsystems unstable,'' *ISA Trans.*, vol. 87, pp. 28–33, Apr. 2019.
[55] P. Kumari and T. Anjali, ''Symmetric-key generation protocol (SGenP) for body sensor network,'' in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6.
[56] F. Sivrikaya and B. Yener, ''Time synchronization in sensor networks: A survey,'' *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, Jul. 2004.
[57] M. Sami Fadali and A. Visioli, *Digital Control Engineering: Analysis and Design*. 2nd ed. Boston, MA, USA: Academic, 2013.