Design and Deployment of Resilient Control Execution Patterns: A Prediction, Mitigation Approach*

Ipsita Koley†, Sunandan Adhikary†, Arkaprava Sain, Soumyajit Dey
Dept. of CSE, Indian Institute of Technology Kharagpur, India
ipsitakoley@iitkgp.ac.in,{mesunandan,arkapravasain}@kgpian.iitkgp.ac.in,soumya@cse.iitkgp.ac.in

ABSTRACT

Modern Cyber-Physical Systems (CPSs) are often designed as networked, software-based controller implementations which have been found to be vulnerable to network-level and physical-level attacks. A number of research works have proposed CPS-specific attack detection schemes as well as techniques for attack-resilient controller design. However, such schemes also incur platform-level overheads. In this regard, some recent works have leveraged the use of skips in control execution to enhance the resilience of a CPS against false data injection (FDI) attacks. However, skipping the control executions may degrade the performance of the controller.

In this paper, we provide an analytical discussion on when and how skipping a control execution can improve the system's resilience against FDI attacks while maintaining the control performance requirement. Our proposed method i) synthesizes a library of such optimal control execution patterns offline, and ii) executes one of them in run-time judging the intent of the attacker. To the best of our knowledge, no previous work has provided any quantitative analysis about the trade-off between attack resilience and control performance for such aperiodic control execution. Finally, we evaluate the proposed method on several safety-critical CPS benchmarks.

CCS CONCEPTS

• **Security and privacy → Embedded systems security**.

KEYWORDS

CPS, control execution skips, security, attack resilient system, control performance

ACM Reference Format:
Ipsita Koley†, Sunandan Adhikary†, Arkaprava Sain, Soumyajit Dey. 2023. Design and Deployment of Resilient Control Execution Patterns: A Prediction, Mitigation Approach. In *ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023) (ICCPS '23), May 9–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3576841.3585922

*The authors acknowledge the generous grants received from "IHUB NTIHAC Foundation - IIT Kanpur" and "Meity Grant No. AAA.22/8/2021-CSRD-Meity" for partially supporting this work. † Authors have equal contributions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICCPS '23, May 9–12, 2023, San Antonio, TX, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0036-1/23/05…$15.00
https://doi.org/10.1145/3576841.3585922

1 INTRODUCTION

Deployment of network components in cyber-physical systems (CPSs) along with software-based, sophisticated control implementations have found wide applicability ranging from industrial control, connected-mobility to defense installations. However, such advancements have opened up different possible attack surfaces leading to network-level as well as physical-level attacks [3]. In this work, we consider a type of attack called *false data injection (FDI)*. Network-controlled CPSs are designed as a closed-loop where the controller (along with an estimator, like Kalman filter, if all states of the plant are not measurable) receives the measurements from the plant, computes the control signal such that the plant operates at/near the desired reference point, and sends the control signal to the plant. As the closed-loop communication happens over a network, an FDI attacker (external or internal) can malign the sensor measurements and the control signals physically or through the network. In consequence, the controller and the plant do not receive actual data, leading to instability or performance loss.

A possible detection and defense mechanism against FDI attacks is to secure each communication using cryptography methods (like using message authentication codes). However, the associated computation and communication overheads incurred by these methods [8, 11] make them infeasible. Alternatively, residue-based lightweight attack detectors [7, 10, 16] can be deployed where statistical methods are applied on the residue (difference between actual and estimated plant outputs) to detect an attack attempt. To detect an FDI attack with high probability, these lightweight detectors take some time. This makes it possible for a smart attacker to intelligently craft *stealthy* FDI attacks which target to maximally degrade the system performance while bypassing the residue-based attack detectors [16]. Thus, the question that arises in this context is *how to make the system more resilient against stealthy FDI attacks?* In this paper, we intend to address this concern.

Generally, mathematical control laws are designed by control designers keeping the environment and platform delays in mind. So, the control loop is designed to attain the desired performance guarantee even in presence of a bounded number of delay-induced control task execution skips [4]. Skipping or dropping a control execution at a certain sampling instance means no new control input will be computed or communicated at that instance, thereby keeping the processor and the communication channel free. As a result, this strategy of skipping certain control executions periodically (thus generating a *skip pattern*) was initially studied to accommodate multiple tasks on resource-constrained embedded platforms [4, 9, 15]. The authors of [1] utilized the effect of such intentional skipping of control executions for a different goal. Their work established how control execution skips can help secure a control loop by increasing the attack effort required to make the

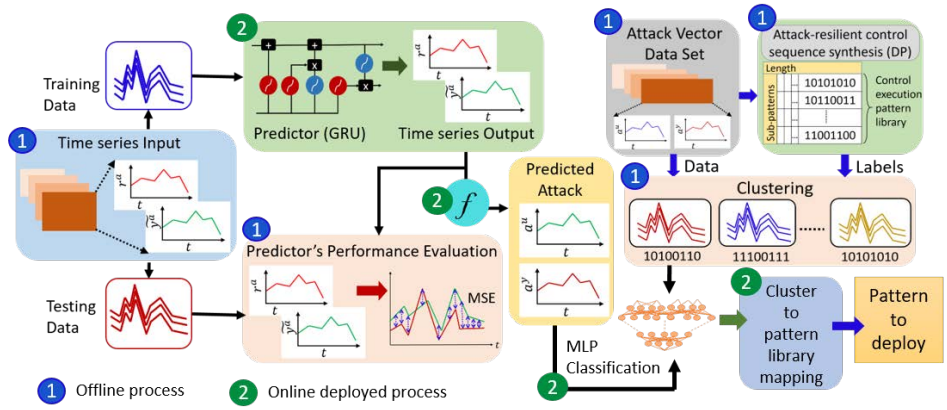


Figure 1: Framework for attack-resilient control sequence synthesis

system unsafe. This is motivated by the fact that the FDIs during such control skips are rendered ineffective since the communication channel is unused during a skip. Hence, a suitable choice of execution skips will increase the minimum number of consecutive measurement/actuation packets that need to be attacked for rendering the system unsafe without getting detected. However, there are major limitations to their approach. *Firstly*, it does not analyze skipping which control execution instances would be effective in terms of increasing the resilience of a system under attack. *Secondly*, their methodology for both attack vector and control execution pattern synthesis does not scale for higher-order systems.

To address such drawbacks in the state-of-the-art, we provide a novel approach that analytically explores the effect of control execution skips/drops on the resilience of any linear dynamical system against FDI attacks. Such an analysis allows us to judiciously choose the control execution instances to skip such that the system becomes more resilient against FDI attacks while providing the desired performance as well by preserving a *minimum rate of control execution* [4]. To this end, we now summarize the contributions of this work as follows.

(1) Attack Effect Analysis: We theoretically derive under which criteria the control execution skips will actually be favourable in enhancing the system's resilience against FDI attacks.

(2) Resilient Control Skip Pattern Synthesis: Utilizing the conditions established in the previous contribution, we design a dynamic programming (DP) based solution methodology to synthesize the control execution patterns that ensure *desired control performance* as well as the *best possible attack-resilience* against a given attack vector. However, in a deployed system, the attack needs to be predicted and a suitable resilient control execution sequence needs to be deployed before an attack actually occurs. This motivates the following contributions.

(3) Attacker Modeling and Attack Generation: Given the specifications of a safety-critical CPS, its initial and safe operating regions, we formulate a constraint-solving problem to generate a library of attack vectors/sequences to model the intent of an FDI attacker who launches optimal or worst-case FDI attack sequences that consume minimum time to make the system unsafe.

(4) Predictor Design: We utilize the generated data set to design a *lightweight predictor* using *Gated Recurrent Units (GRU)* that would predict the most probable attack scenario over a window of sampling instances in the future given a window of observations in past. We train the predictor to learn an attacker's intent by observing the system parameters under attack.

(5) Runtime Classification and Deployment: We form clusters of the observed attack vectors for which the optimal attack-resilient control execution patterns (generated by applying contribution 2) are the same. This forms our basis for clustering attack vectors to a finite number of clusters where each cluster is labeled with a candidate control execution pattern. Given this set of clusters, we design a *multi-layer perceptron (MLP)*-based *lightweight classifier* which takes as input a predicted attack vector (from contribution 4) and determines which cluster it belongs to. The corresponding optimal attack-resilient control execution pattern is deployed in the system. Thus, contributions 4 and 5 execute in a loop continuously in the system at runtime with suitable periodicity.

(6) Evaluation: The scalability of the proposed method has been evaluated on a hardware-in-loop (HIL) based real-time PC setup using well-known benchmarks with various dimensions.

Fig. 1 nicely captures how the highlighted contributions step by step build our overall methodology.

2 BACKGROUND

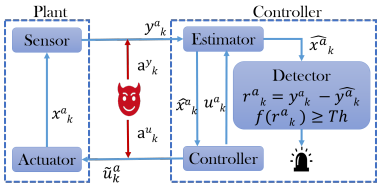


Figure 2: Secure CPS architecture

Secure CPS Model: The general architecture of a secure CPS is presented in Fig. 2. The physical process, i.e., the plant and the controller work together in a closed-loop manner such that the desired operating criterion of the physical process is maintained. They communicate between themselves over a network, which we consider is vulnerable to FDI attacks. In the absence of an adversary, the closed-loop dynamics of a CPS can be presented as a discrete linear time-invariant (LTI) system like the following.

$$x_{k+1} = Ax_k + Bu_k + w_k;\ y_k = Cx_k + v_k;\ \hat{y}_{k+1} = C(A\hat{x}_k + Bu_k);$$
$$r_{k+1} = y_{k+1} - \hat{y}_{k+1};\ \hat{x}_{k+1} = A\hat{x}_k + Bu_k + Lr_{k+1};\ u_k = -K\hat{x}_k;\ e_k = x_k - \hat{x}_k \quad (1)$$

Here, $x_k \in \mathbb{R}^n$ is the system state vector, $y_k \in \mathbb{R}^m$ is the measurement vector obtained from available sensors at k-th time stamp; A, B, C are the system matrices. We consider that the initial state $x_0 \in \mathcal{N}(\bar{x}_0, \Sigma)$, the process noise $w_k \in \mathbb{R}^n \sim \mathcal{N}(0, \Sigma_w)$ and the measurement noise $v_k \in \mathbb{R}^m \sim \mathcal{N}(0, \Sigma_v)$ are independent Gaussian random variables. Further, in every k-th sampling instant, the observable system state $\hat{x}_k$ is estimated using system output y_k while minimizing the effect of noise, and used for computing the control input $u_k \in \mathbb{R}^l$. The estimation error e_k is defined as the difference between actual system states x_k and estimated system states $\hat{x}_k$. We denote the residue, i.e., the difference between the measured and the estimated outputs as r_k. The estimator gain L and controller gain K are designed in such a way that it is ensured both $(A - LC)$ and $(A - BK)$ are stable.

Consider an FDI attack, where the attacker injects false data a_k^y and a_k^u (Fig. 2) to the sensor measurement and control signal respectively. In this case, the system's dynamical equation becomes,

$$x^a_{k+1} = Ax^a_k + B\tilde{u}^a_k + w_k;\ y^a_k = Cx^a_k + v_k + a^y_k$$
$$\hat{y}^a_{k+1} = C(A\hat{x}^a_k + Bu^a_k);\ r^a_{k+1} = y^a_{k+1} - \hat{y}^a_{k+1}$$
$$\hat{x}^a_{k+1} = A\hat{x}^a_k + Bu^a_k + Lr^a_{k+1}; u^a_k = -K\hat{x}^a_k;\ \tilde{u}^a_k = u^a_k + a^u_k;\ e^a_k = x^a_k - \hat{x}^a_k; \quad (2)$$

Here, x^a_k, $\hat{x}^a_k$, y^a_k, r^a_k, u^a_k, $\tilde{u}^a_k$, and e^a_k represent plant state, estimated plant state, forged sensor data, residue, control signal, forged control signal, and estimation error respectively in an FDI attack scenario. u^a_k is the control input computed at k-th sampling instance on which the effect previous attacks i.e. a^u_i and a^y_i for $1 \le i \le k-1$ persist. u^a_k added with attack on actuator a^u_k at k-th sample produces $\tilde{u}^a_k$, i.e., the forged control signal at k-th sample. Note that even though we discussed intrusion through the network, physical level sensor data tampering [14] can also happen. The above attack model is generic to all kinds of such falsification attacks. The objective of such attacks is to violate the *safety property* of the system. We define the safety property as $x^a_k \in X_S$, $\forall k > 0$ where X_S is the safety envelope of the system. As security enforcement, we consider a residue-based detector as demonstrated in Fig. 2. The residue-based detector computes a function $f(r_k)$ (f can be a simple norm or any statistical method, like χ^2-test) and compares it with a threshold Th to identify any anomalous behavior of the system. We denote an attack vector at k-th sampling instance as $\mathbf{a}[k] = [a^u_k, a^y_k]^T$. If the attacker continues the false data injection for l sampling iterations, then the l length attack vector is expressed as follows $\mathbf{a}_l = [\mathbf{a}[1] \cdots \mathbf{a}[l]] = \begin{bmatrix} a^u_1 & \cdots & a^u_l \\ a^y_1 & \cdots & a^y_l \end{bmatrix}$. Falsifying the control input by injecting a sequence of a^u's, the attacker forces the states of the system to go beyond the safety envelope X_S. On the other hand, it modifies the sensor measurements with a sequence of a^y's such that it can hide from the residue-based detector. We define such *successful and stealthy* FDI attack vector as follows.

Definition 1 (**Successful and Stealthy false data injection attack**). *An attack vector* $\mathbf{a}_l = [\mathbf{a}[1] \cdots \mathbf{a}[l]] = \begin{bmatrix} a^u_1 & \cdots & a^u_l \\ a^y_1 & \cdots & a^y_l \end{bmatrix}$ *of length* l *is said to be stealthy if* $f(r^a_k) < Th\ \forall k \in [1, l]$, *and successful if* $\exists k \in [1, l]$ *s.t.* $x^a_k \notin X_S$. □

Control Execution Skip Pattern: When we say that a control execution is skipped in a certain k-th sampling instance, the implication of the same on the underlying system is as follows. *(1)* The sensor measurements y_k are not communicated to the controller and the estimator units. *(2)* A fresh control input u_k is not calculated and communicated to the plant. The plant and the estimator update their states simply using the previous control input. *(3)* Detection unit will also not operate. We consider that such skips in control execution shall lead to a *control execution skip pattern* (in lines of [4]) which is formally defined below.

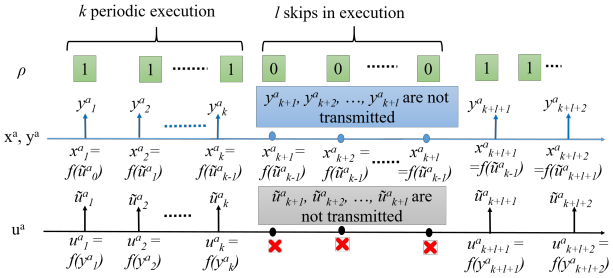


Figure 3: Implication of control skip pattern on the system

Definition 2 (**Control Execution Skip Pattern**). *A t length control execution skip pattern for a given control loop (A, B, C, K, L), is a sequence $\rho \in \{0, 1\}^t$ such that it can be used to define an infinite length control execution sequence π, repeating after every t samples, defined as, $\pi[k] = \pi[k + t] = \rho[k\%t], \forall k \in \mathbb{Z}^+$ where A, B, and C are system matrices, K is the controller gain, and L is the observer gain of the system. [4].* □

In a pattern ρ, 1 denotes control execution and 0 denotes control execution skip. Some examples of patterns are 1^t = (periodic execution, i.e., 0 skip), $(10)^t = 101010\cdots$, $111001010\cdots$, etc. In the remaining part of the paper, we refer *periodic execution* when no control execution is skipped and *aperiodic execution* when at least one control execution is skipped. Consider that there is a skip in control execution at $(k+1)$-th sample as demonstrated in Fig. 3. Then according to skip properties, $u_k = u_{k-1}$, $a^u_k = a^u_{k-1}$, and $\triangle r_{k+1} = 0$. Therefore, plant state x_{k+1} at $(k+1)$-th sample will be updated by old control input, i.e., u_{k-1}. The notion of skip changes the system dynamics like the following.

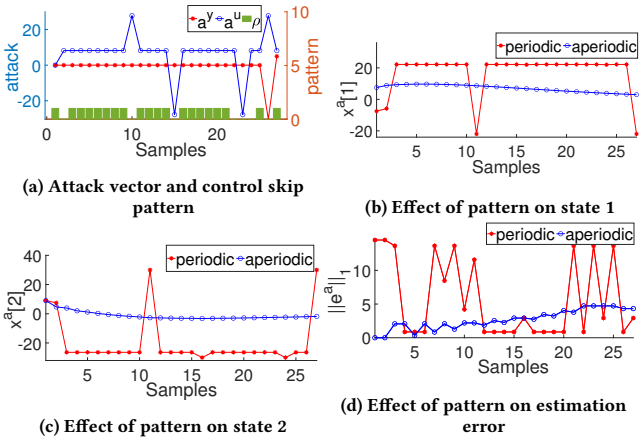
$$x^a_{k+1} = Ax^a_k + B(u^a_{k-1} + a^u_{k-1}) + w_k;\ \hat{x}^a_{k+1} = A\hat{x}^a_k + Bu^a_{k-1};$$
$$x_{k+1} = Ax_k + Bu_{k-1} + w_k;\ \hat{x}_{k+1} = A\hat{x}_k + Bu_{k-1};$$
$$e^a_{k+1} = Ae^a_k + Ba^u_{k-1} + w_k;\ e_{k+1} = Ae_k + w_k; \quad (3)$$

Control Performance: In this work, we define the control performance with respect to the settling time T_s of a system. Settling time is the duration within which the system output must reach and stay within a specified percentage of the reference. To ensure the desired performance while some of the control executions are skipped, the control execution must maintain the minimum rate r_{min} [4] such that the settling time property can be achieved. For example, in a window of t samples, the controller must be executed $\lceil t \times r_{min} \rceil$ times. This implies in a control execution skip pattern ρ of length t, there must be at least $\lceil t \times r_{min} \rceil$ 1's.

3 ANALYSING EXECUTION SKIPS

A Motivating example: We demonstrate how occasional skips in control execution improve the system's resilience against FDI

attacks with help of a motivating example (Fig. 4). We take the example of a trajectory tracking control (TTC) system of a vehicle from [1]. This is a 2-dimensional system with the *deviation* from the reference trajectory and *velocity* of the vehicle as states. The attacker adds false data to the measurement data (i.e., deviation) and the control signal (i.e., acceleration). In Fig. 4a, the attacks on



(a) Attack vector and control skip pattern

(b) Effect of pattern on state 1

(c) Effect of pattern on state 2

(d) Effect of pattern on estimation error

Figure 4: Demonstrating the effect of control skip pattern on system's resilience against FDI attacks

measurement a^y and control signal a^u are given. We design a control skip pattern ρ by introducing drops in control executions with the intention to weaken the attack's effect. Thus, we inject the skip where the attack on actuation $||a^u||$ increases (green bar graph in Fig. 4a). The minimum execution rate of the controller is also more than the requirement, i.e., 50% [1]. The effect of introducing drops in the presence of the attacker is presented in Fig. 4b-4d. We can observe that when the attacker injects optimal attack values (Fig. 4a), there is a significant deviation in the system's state progression (Fig. 4b and 4c) when periodic control execution takes place. However, due to the introduction of drops in the control execution, FDIs rendered ineffective on the system's state progression (Fig. 4b and 4c) in case of aperiodic control execution. This is because the FDIs in periodic execution increase the estimation error e^a considerably (Fig. 4d). This in turn affects the control performance poorly (according to Eq. 2). On the other hand, ignorance of the attack values on actuation signal during drops minimizes the attack's effect on e^a, thereby containing e^a within a much lower range (Fig. 4d). Thus, we can see the aperiodic control execution following the pattern ρ (Fig. 4a) enhances TTC's resilience against the FDI attack given in Fig. 4a. Keeping in mind the stealthy FDI attack's effect on the system's state progression, how to generate the pattern to weaken the attack effect most effectively, is discussed in Sec. 4. Now, we provide a quantitative analysis behind such motivation of this work.

Attack Induced Estimation Error Analysis: During execution skips, the attacks on sensor data and control signals are ignored. This seems useful with respect to attack resiliency. But, during control execution skips, no new control input is computed and communicated to the plant. The plant updates its states using the previous control input. This may lead to poor control performance. Naturally, there exists a trade-off between resilience against the attack and the amount of control performance to forego during skips. Our aim is to provide a formal discussion on when and how skips can improve resilience against FDI attacks and at the same time desirable control performance is maintained when the control execution is aperiodic. First, to capture the difference in the system's response in the presence and absence of FDI attacks in case of periodic control execution, we introduce the following two terms:

$$\begin{aligned}\triangle e_k = e^a_k - e_k &= (A - LCA)\triangle e_k + (B - LCB)a^u_k - La^y_{k+1} \\ &= A\triangle e_k + Ba^u_k - L\triangle r_{k+1} = \sum_{i=0}^{k-1} A^i(Ba^u_{k-1-i} - L\triangle r_{k-i}) \ (\triangle e_0 = a^y_0 = 0)\end{aligned} \tag{4}$$

$$\triangle r_k = r^a_k - r_k = CA\triangle e_k + CBa^u_k + a^y_{k+1} \tag{5}$$

Here, $\triangle e$ and $\triangle r$ present how much the estimation error and residue vary due to the FDI attack. Let us consider, there is a skip in control execution at $(k+1)$-th sample. Then, following Eq. 3, we get,

$$\begin{aligned}\triangle e_{k+1} = e^a_{k+1} - e_{k+1} &= (Ae^a_k + Ba^u_{k-1} + w_k) - (Ae_k + w_k); \\ &= A\triangle e_k + Ba^u_{k-1}\end{aligned} \tag{6}$$

$$\triangle r_{k+1} = 0 \tag{7}$$

Now, to show whether the execution skips actually enhance the resilience of the system against FDI attacks, we compare two parameters: i) $\triangle e^p_{k+l}$, i.e., estimation error deviation after $(k+l)$ periodic executions and ii) $\triangle e^{ap}_{k+l}$, i.e., estimation error deviation after k periodic executions followed by l control execution drops. This is demonstrated in Fig. 3. The control execution is periodic during 1st k samples. The closed-loop system progresses following Eq. 2. At $(k+1)$-th sample when the control execution is dropped, measurement of that sampling instance y^a_k is not transmitted, the control input u^a_k is neither computed nor transmitted, and the state progresses with the last received control signal $\tilde{u}^a_k$ (Eq. 3). This is continued till $(k+l)$-th sample. Once, the controller receives new measurement at $(k+l+1)$-th sample, the sensor measurement y^a_{k+l+1} is transmitted to the plant, the control input u^a_{k+l+1} is again computed using y^a_{k+l+1} (Eq. 2) and transmitted to the plant. The plant updates its states following Eq. 2 using new control input. Now, from Eqs. 4 and 6, we capture the iterative forms of $\triangle e^p_{k+l}$ and $\triangle e^{ap}_{k+l}$ as,

$$\begin{aligned}\triangle e^p_{k+l} &= A\triangle e_{k+l-1} + Ba^u_{k+l-1} - L\triangle r_{k+l} \\ &= A[A\triangle e_{k+l-2} + Ba^u_{k+l-2} - L\triangle r_{k+l-1}] + Ba^u_{k+l-1} - L\triangle r_{k+l} \\ &= A^2\triangle e_{k+l-2} + (ABa^u_{k+l-2} + Ba^u_{k+l-1}) - (AL\triangle r_{k+l-1} + L\triangle r_{k+l}) \\ &= \cdots \\ &= A^l\triangle e_k + (A^{l-1}Ba^u_k + A^{l-2}Ba^u_{k+1} + \cdots + Ba^u_{k+l-1}) \\ &\quad - (A^{l-1}L\triangle r_{k+1} + A^{l-2}L\triangle r_{k+2} + \cdots + L\triangle r_{k+l}) \\ &= A^l\triangle e_k + \sum_{i=0}^{l-1} A^i(Ba^u_{k+l-1-i} - L\triangle r_{k+l-i})\end{aligned} \tag{8}$$

$$\begin{aligned}\triangle e^{ap}_{k+l} &= A\triangle e_{k+l-1} + Ba^u_{k-1} = A[A\triangle e_{k+l-2} + Ba^u_{k-1}] + Ba^u_{k-1} \\ &= A^2\triangle e_{k+l-2} + (A+I)Ba^u_{k-1} \\ &= \cdots \\ &= A^l\triangle e_k + (A^{l-1} + A^{l-2} + \cdots + 1)Ba^u_{k-1} = A^l\triangle e_k + \sum_{i=0}^{l-1} A^iBa^u_{k-1}\end{aligned} \tag{9}$$

The first terms in both Eq. 8 and 9 represent estimation error deviation up to k-th sampling instance. In both periodic and aperiodic cases, this term will be the same. And, the second term in Eq. 8 and 9 captures the effect of the last l iterations in periodic and

aperiodic cases respectively. We define the *resilience* of a system against FDI attacks with respect to the terms $\triangle e^p$ and $\triangle e^{ap}$. If the FDI attack fails to do much harm to the system, the values of $\triangle e^p$ and $\triangle e^{ap}$ will be less. Thus, lower values of $\triangle e^p$ and $\triangle e^{ap}$ imply that the system is more resilient against FDI attacks. Occasional skips in control execution will be useful in enhancing the system's resilience against FDI attacks if the difference in estimation error in periodic execution due to attack $\triangle e^p$ is more than that of aperiodic control execution $\triangle e^{ap}$. In the following theorem (proof is given in Appendix), we establish under which condition $\triangle e^p$ will be more than $\triangle e^{ap}$.

Theorem 1. *For a plant-controller closed-loop system under FDI attack (Eq. 2), control execution skips for consecutive l sampling instances after k periodic control executions will be effective in enhancing the system's resilience when the following criterion is true:* $||\sum_{i=0}^{l-1} A^i B \triangle a^u_{k+l-1-i}|| > ||\sum_{i=0}^{l-1} L \triangle r_{k+l-i}||$
Here, the term $\triangle a^u_{k+l-1-i} = a^u_{k+l-1-i} - a^u_{k-1}, \forall i \in [0, l-1]$ captures the difference between the attacks on control signal on the last l sampling instances out of $(k+l)$ samples between periodic (i.e., none of the $k+l$ samples has been skipped) and aperiodic cases (i.e., last l of the $k+l$ samples has been skipped). □

Remark 1. *For an aperiodic control execution pattern of k consecutive periodic execution followed by l control skips, the effect of actuation attack at $(k-1)$-th sample (starting from the 0-th sample), i.e., a^u_{k-1} gets forwarded through all of the next $(l-1)$ iterations. Whereas, in the case of periodic control executions, an attacker can inject different actuation attack values $a^u_{k+l-1-i}$ at each of the sampling iterations, i.e., $\forall i \in [0, l-1]$. The above theorem states that whenever the attacker attempts to vary the attack efforts in consecutive iterations in order to stay stealthy or jeopardize the system safety faster (i.e., $||\sum_{i=0}^{l-1} A^i B \triangle a^u_{k+l-1-i}|| \neq 0$), skipping the control execution at that sampling instances helps to make the system more resilient against the injected false data.* □

4 ATTACK-RESILIENT CONTROL EXECUTION SEQUENCE SYNTHESIS

In this section, we thoroughly discuss the steps to synthesize the control execution skip patterns that exhibit optimal resilience against stealthy FDI attacks utilizing the criteria derived in the last section. We generate control skip patterns in 2 steps. *First,* utilizing the condition presented in Theorem 1, we generate a list of t length *sub-patterns* that are beneficial in enhancing the system's resilience against FDI attacks. In the *second* step, we formulate a DP-based solution method to compute the final, i.e., optimal attack-resilient control execution skip pattern by merging the sub-patterns generated in the first step. The DP-based formulation also facilitates generating a list of control execution skip patterns ranked in order of the advantage metric. We now elaborately discuss these 2 steps.

Step 1: Favourable Sub-pattern synthesis: We denote a t-length sub-pattern as a binary string of the form $\rho(k,l) = 1^k 0^{l-k} 1^{t-l}$ where $k > 0$ and $l \geq 0$. This implies periodic execution of the controller in the first k iterations, followed by execution skips till l-th iteration and then $(t-l)$ periodic executions. How much a sub-pattern $\rho(k,l)$ is beneficial in enhancing the system's resilience against stealthy FDI attacks, is quantified with an *advantage* value as defined next.

Require: State matrices A, B and C, controller gain K, observer gain L, sensor limit $\mathcal{Y}$, actuation saturation limit $\mathcal{U}$, detector threshold Th, safety envelope X_s, initial region X_0 of the plant states, minimum execution rate r_{min} of the controller, an attack vector $\mathbf{a}$
Ensure: List of favourable sub-patterns *subPatternList* and their advantage metric D
1: **function** AdvPatSyn($A, B, C, K, L, \mathcal{Y}, \mathcal{U}, Th, X_s, X_0, r_{min}, \mathbf{a}$)
2: $D[i][i] \leftarrow 0\ \forall i \in [1,t]$; *subPatternList* $\leftarrow$ *null*; ▷ Initialization
3: $t \leftarrow length(\mathbf{a})$;
4: $\triangle r \leftarrow$ ResDiffGen($A, B, C, K, L, X_0, \mathbf{a}, t$); ▷ $\triangle r[i] = r^a_i - r_i$ (Eq. 5)
5: **for** k=1 to t **do**
6: **for** l=1 to t-k **do**
7: $lhs \leftarrow 0$; $rhs \leftarrow 0$; $\rho(k,l) \leftarrow 1^t$;
8: **for** i=1 to l **do**
9: **if** k>1 **then** $lhs \leftarrow lhs + A^{i-1} B (a^u_{k+l-i} - a^u_{k-1})$;
10: **else** $lhs \leftarrow lhs + A^{i-1} B a^u_{k+l-i}$; ▷ We assume $a^u_0 = 0$
11: $rhs \leftarrow rhs + A^{i-1} L \triangle r[k+l-i+1]$;
12: **if** $||lhs|| > ||rhs||$ **then**
13: $\rho(k,l) \leftarrow 1^k 0^{l-k} 1^{t-l}$;
14: **if** $sum(\rho(k,l)) \geq r_{min} \times t$ **then** $D[k][k+l] \leftarrow ||lhs|| - ||rhs||$;
15: **else** $\rho(k,l) \leftarrow 1^t$;
16: $subPatternList \leftarrow subPatternList \cup \rho(k,l)$;
17: **return** D, *subPatternList*;

Algorithm 1: Favourable Sub-pattern Synthesis

Definition 3 (**Advantage value of a sub-pattern**). *The advantage of control execution that follows sub-pattern $\rho(k,l) = 1^k 0^{l-k} 1^{t-l}$ of length t over periodic control execution of length t (i.e., 1^t) is quantified by the value $||\triangle e^p|| - ||\triangle e^{ap}|| = ||\sum_{i=0}^{l-1} A^i B \triangle a^u_{k+l-1-i}|| - ||\sum_{i=0}^{l-1} A^i L \triangle r_{k+l-i}||$ (Theorem 1).* □

For a given system specification $\langle A, B, C, K, L, X_0, X_s \rangle$, where A, B, and C are system matrices, K and L are controller and estimator gains, X_0 and X_s are the initial and safety regions of the plant, we need to find the attack-resilient control execution sub-pattern $\rho = 1^k 0^l 1^{t-l}$ of length t where $k > 0, l \geq 0$ such that the advantage value $||\sum_{i=0}^{l-1} A^i B \triangle a^u_{k+l-1-i}|| - ||\sum_{i=0}^{l-1} A^i L \triangle r_{k+l-i}|| > 0$, provided the minimum execution rate r_{min} of the controller is maintained.

We present a method to synthesize a list *subPatternList* of such favourable sub-patterns of the form $\rho(k,l)$ in Algo. 1 which also stores the advantage value of the sub-patterns in a matrix D of size $t \times t$. $D[k][l]$ contains the advantage value of $\rho(k,l)$. The following are provided as input to the algorithm: i) system matrices A, B, and C, controller gain K, observer gain L, ii) the maximum limit $\mathcal{Y}$ of the sensor measurements, actuation saturation limit $\mathcal{U}$, iii) the threshold Th of the detector in place, iv) performance criteria of the controller, i.e., minimum execution rate r_{min}, v) the safety property of the system, defined as a safety polytope X_s which mandates that the system trajectory will always be within X_s, vi) initial region of the plant states $X_0 \in \mathbb{R}^n$ from which the system progression initiates, and vii) an attack vector $\mathbf{a}$ against which the resilient control skipping patterns are to be generated. In line 2, we initialize the advantage matrix D with 0 and *subPatternList* as null. We store the length t of the input attack vector $\mathbf{a}$ in line 3. By simulating the system's state progression under no attack and under the attack $\mathbf{a}$ in a periodic control execution for t iterations (Eq. 1 and 2), we compute $\triangle r^a_i$ for all $i \in [1,t]$ and store them in the array $\triangle r$ (line 4). The for loop in line 5 signifies the possible number of consecutive 1's in the sub-pattern $\rho(k,l)$ and the for loop in line 6 signifies the number of consecutive 0's following the consecutive 1's in $\rho(k,l)$. The LHS and RHS of the criteria $||\sum_{i=0}^{l-1} A^i B \triangle a^u_{k+l-1-i}|| > ||\sum_{i=0}^{l-1} A^i L \triangle r_{k+l-i}||$ (Theorem 1) are computed in lines 9-10 and line 11 respectively.

Matrix D

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0.02	0.08	0.15	3.7	0	0
3	0	0	0	0.05	0.12	3.68	1.67	0
4	0	0	0	0	0.06	3.6	1.64	1.77
5	0	0	0	0	0	3.52	1.56	1.72
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0.08
8	0	0	0	0	0	0	0	0

Matrix D: *D[k][l]* contains the advantage value of the sub-pattern $\rho(k,l)$
Matrix M: *M[i][j]* contains the maximum advantage gained till length j considering first i sub-patterns
Matrix P: *P[i][j]* contains j length optimal attack-resilient control execution pattern considering first i sub-patterns

Matrix M

	1	2	3	4	5	6	7	8
1: ρ(2,3)	0	0	0.02	0.02	0.02	0.02	0.02	0.02
2: ρ(2,4)	0	0	0.02	0.08	0.08	0.08	0.08	0.08
3: ρ(2,5)	0	0	0.02	0.08	0.08	0.15	0.15	0.15
4: ρ(2,6)	0	0	0.02	0.08	0.08	0.15	0.15	3.7
5: ρ(3,4)	0	0	0.02	0.08	0.08	0.15	0.15	3.7
6: ρ(3,5)	0	0	0.02	0.08	0.12	0.15	0.15	3.7
7: ρ(3,6)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
8: ρ(3,7)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
9: ρ(4,5)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
10: ρ(4,6)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
11: ρ(4,7)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
12: ρ(4,8)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
13: ρ(5,6)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
14: ρ(5,7)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
15: ρ(5,8)	0	0	0.02	0.08	0.12	3.68	3.68	3.7
16: ρ(7,8)	0	0	0.02	0.08	0.12	3.68	3.68	3.76

Matrix P

	1	2	3	4	5	6	7	8
1: ρ(2,3)	11111111	11111111	11011111	11011111	11011111	11011111	11011111	11011111
2: ρ(2,4)	11111111	11111111	11011111	11001111	11001111	11001111	11001111	11001111
3: ρ(2,5)	11111111	11111111	11011111	11001111	11001111	11000111	11000111	11000111
4: ρ(2,6)	11111111	11111111	11011111	11001111	11001111	11000111	11000111	11000011
5: ρ(3,4)	11111111	11111111	11011111	11001111	11001111	11000111	11000111	11000011
6: ρ(3,5)	11111111	11111111	11011111	11001111	11100111	11000111	11000111	11000011
7: ρ(3,6)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
8: ρ(3,7)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
9: ρ(4,5)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
10: ρ(4,6)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
11: ρ(4,7)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
12: ρ(4,8)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
13: ρ(5,6)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
14: ρ(5,7)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
15: ρ(5,8)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
16: ρ(7,8)	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11100010

Figure 5: Optimal attack-resilient pattern generation using dynamic programming approach

In line 12, we check if the LHS is more than the RHS, i.e., if the difference in estimation error under periodic execution $\triangle e^p$ is more than that of aperiodic execution $\triangle e^{ap}$. If yes, we generate a new sub-pattern $\rho(k,l)$ by introducing skips from $(k+1)$ to $(k+l)$ in line 13. Next, in line 14, we update the advantage value matrix D with $||lhs|| - ||rhs||$ (which is nothing but $(||\triangle e^p|| - ||\triangle e^{ap}||)$) if the sub-pattern $\rho(k,l)$ satisfies the minimum execution rate condition. Else, we modify the sub-pattern $\rho(k,l)$ as a periodic one (line 15). Next, we include $\rho(k,l)$ (periodic or aperiodic) into *subPatternList* (line 16). Finally, the algorithm returns the list of t length sub-patterns of the form $\rho(k,l)$ along with the matrix D which contains the advantage values of those sub-patterns (line 17). The time complexity of Algo. 1 is $O(t^3)$.

Step 2: Optimal Attack-resilient Pattern Synthesis: We now formulate a dynamic programming (DP) based method to synthesize the optimal attack-resilient control execution patterns using the *subPatternList* and D generated from Algo. 1. We demonstrate the method with help of the example given in Fig. 5 where the length of the input attack vector **a** is $t = 8$ and the minimum rate criteria $r_{min} = 0.5$. In the DP formulation, we maintain 2 matrices: $M_{p\times t}$ and $P_{p\times t}$ where t and p are respectively the length of **a** and the number of sub-patterns in *subPatternList*. For each sub-pattern $\rho(k,l) \in subPatternList$, $D[k][l]$ has a non-trivial entry. The row indices of M, P, and the *subPatternList* are basically lexicographic ordering of such non-trivial (k,l)-pairs. For example, since $D[2][3], \cdots, D[2][6]$ are non-trivial, the first four rows of M, P are $1 : \rho(2,3), \cdots, 4 : \rho(2,6)$ in Fig. 5. The maximum advantage value (Def. 3) that can be achieved by considering skips till j-th position in the pattern of length t, i.e. $\{0,1\}^t$, when only the first i sub-patterns in the *subPatternList* are taken into consideration, is computed and stored in $M[i][j]$. The corresponding optimal pattern is stored in $P[i][j]$.

Let the i-th sub-pattern be $\rho(k,l) = 1^k 0^{l-k} 1^{t-l}$. For this, let $k = end1(i)$ and $l = end0(i)$ denote the indices where the initial 1's and 0's of the sub-pattern $\rho(k,l)$ finish. The rate of control execution for any t-length pattern containing a total of n 1s is given by $rate = n/t$. We define the merging of two patterns i and j using element-wise logical AND operation and denote the merged pattern by $i \circ j$. With help of the example in Fig. 5, we now elaborate on how to populate M (Eq. 10) and P using DP memoization process.

case 1 ($i = 1$): The 1-st sub-pattern in the list *subPatternList* is used to populate first row of M and P. Since we do not have any favourable sub-pattern with skips up to the length $(end0(1) - 1)$ with non-zero advantage value, we populate $M[1][j] = 0$ and $P[1][j] = 1^t$ for $j < end0(1)$. Let us consider the first row of M and P in the example of Fig. 5. The first rows of M and P in the example of Fig. 5 correspond to the sub-pattern $\rho(2,3)$. Consider the first $end0(1)-1 = 2$ columns of M. The sub-pattern $\rho(2,3)$ is of the form $1^2 0^{3-2} 1^{8-3}$ = 11011111, in which there is no prefix up to length 2 (i.e., **[1]**1011111 or **[11]**011111) that produces positive advantage value (refer to matrix D). Therefore, we update $M[1][1] = M[1][2] = 0$ and $P[1][1] = P[1][2] = 1^t$. In the first row, for $j \geq end0(1)$, we populate the matrix M with the advantage value of the first sub-pattern and P with the first sub-pattern if the rate of the first sub-pattern up to length j, i.e., $1^{end1(1)} 0^{end0(1)-end1(1)} 1^{j-end0(1)} = 1^2 01^{j-3}$ satisfies the rate criteria. Otherwise, $M[i][j]$ and $P[i][j]$ are assigned 0 and 1^t respectively. In Fig. 5, since $\rho(2,3)$ satisfies r_{min} up to the length $j \geq end0(1) = 3$, we populate $M[1][j]$ with $D[2][3]$ and $P[1][j]$ with $1^2 0^{3-2} 1^{8-3} = 1^2 01^5 \; \forall j \geq 3$.

	1	2	3	...	8
1: ρ(2,3)	0	0	0.02	...	0.02
2: ρ(2,4)				...	-
⋮	...	...	...	...	...
16: ρ(7,8)	-	-	-	...	-

Figure 6: DP case 2

case 2 ($i > 1$ and $j < end0(i)$): Now, let us consider the other sub-patterns in *subPatternList*. There is no prefix up to $(end0(i) - 1)$ length in the i-th sub-pattern $[\mathbf{1^{end1(i)} 0^{end0(i)-end1(i)-1}}] 01^{t-end0(i)}$ that produces positive advantage value. Therefore, for $i > 1, j < end0(i)$, we can update $M[i][j]$ with $M[i-1][j]$ because $M[i-1][j]$ holds the best advantage value that can be achieved considering previous $(i-1)$ sub-patterns up to length j. Similarly, we assign $P[i][j] = P[i-1][j]$ for $j < end0(i)$. Consider the second row of the matrices M and P in Fig. 5, i.e., the one corresponding to sub-pattern $\rho(2,4) = 1^2 0^{4-2} 1^{8-4}$ = 11001111 (highlighted in red). For, $j < end0(2)$, i.e., $j < 4$, there exists no prefix in 11001111 that yields a positive advantage value. Therefore, we set $M[2][j] = M[1][j]$ (Fig. 6) as $M[1][j]$ holds the maximum advantage that can be gained by considering skips until j ($j < 4$) positions of a $t = 8$ length pattern. Similarly, $P[2][j]$ is assigned with $P[1][j]$.

case 3 ($i > 1$ and $j \geq end0(i)$): We divide this case into 3 scenarios.

(i) If the j-length prefix of the i-th sub-pattern does not satisfy the rate criteria, we set $M[i][j] = M[i-1][j]$ and $P[i][j] = P[i-1][j]$.

	1	...	6	...
1: ρ(2,3)	0	...	0.02	...
2: ρ(2,4)	0	...	0.08	...
3: ρ(2,5)	0	...	0.15	...
4: ρ(2,6)	0	...		...
⋮	-	-	-	...

Figure 7: DP case 3i

Consider the row $i = 4$ (highlighted in blue) and column $j = 6$ of M in Fig. 5. The 4-th row corresponds to the sub-pattern $\rho(2,6) = 1^2 0^{6-2} = 11000011$. In $M[4][6]$, we want to have the maximum advantage value that can be achieved by considering skips until the 6-th position of the 8-length pattern. And, we can compute $M[4][6]$ by considering only the first 4 sub-patterns, i.e., $\rho(2,3)$, $\rho(2,4)$, $\rho(2,5)$ and $\rho(2,6)$. However, the $j = 6$ length prefix of **[110000]**11 does not satisfy $r_{min} = 0.5$. So, we set $M[4][6] = M[3][6]$ and $P[4][6] = P[3][6]$ (Fig. 7).

(ii) If the j-length prefix of the i-th sub-pattern satisfies the rate criteria, we can consider merging the i-th sub-pattern with the most favourable *non-overlapping* sub-patterns. Two sub-patterns are *non-overlapping* if they do not have 0's at the same position. Therefore, the candidate patterns which can be merged with i-th sub-pattern $1^{end1(i)}0^{end0(i)-end1(i)-1}01^{t-end0(i)}$ must have 0's before their $end1(i)$-th position. As per the construction of the P matrix, the most favourable candidate sub-pattern for merging with i-the sub-pattern is stored in $P[i][end1(i)-1]$. However, the i-th sub-pattern can be merged with $P[i][end1(i)-1]$ if the j-length prefix of the merged pattern satisfies minimum rate r_{min} criteria. If the rate condition on the merged pattern is not satisfied, we check which one between the $P[i-1][j]$ and j-length prefix of the i-th sub-pattern (note that j-length prefix of the i-th sub-pattern satisfies rate criteria) yields better advantage. Accordingly, we set $M[i][j] = max\{M[i-1][j], D[end1(i)][end0(i)]\}$ (Fig. 8) and populate $P[i][j]$. Consider the row $i = 12$ (highlighted in yellow) and column $j = 8$ of M in Fig. 5. The 12-th row corresponds to the sub-pattern $\rho(4,8) = 1^4 0^{8-4} 1^0 = 11110000$.

	1	2	...	8
1: ρ(2,3)	0	0	...	0.02
⋮	...	...	...	
11: ρ(4,7)	0	0	...	3.7
12: ρ(4,8)	0	0	...	Max(___, D[4][8])
⋮	...	...	...	...

Figure 8: DP case 3ii

In $M[12][8]$, we want to have the maximum advantage value that can be achieved by considering skips until the last position of the 8-length pattern. And, we can compute $M[12][8]$ by considering only the first 12 sub-patterns, i.e., $\rho(2,3)$, $\rho(2,4)$, $\cdots$, $\rho(4,8)$. The pattern $\rho(4,8)$ satisfies the rate condition and is also mergeable with a non-overlapping sub-pattern in $P[12][3] = 1101111$. However, the merged pattern $P[12][3] \circ \rho(4,8) = 11011111 \circ 11110000 = 11010000$ does not satisfy $r_{min} = 0.5$. Thus, we set $M[12][8] = max\{M[11][8], D[4][8]\} = M[11][8]$ (Fig. 8), and accordingly set $P[12][8] = P[11][8] = 11000011$.

(iii) Finally, consider that the merged pattern of i-th sub-pattern and the pattern in $P[i][end1(i)-1]$ satisfies the rate condition. Then, we check if we can yield a better advantage after merging. If so, we set $M[i][j] = M[i][end1(i)-1] + D[end1(i)][end0(i)]$ and $P[i][j] = P[i][end1(i)-1] \circ 1^{end1(i)}0^{end0(i)-end1(i)}1^{t-end0(i)}$. Otherwise, we assign $M[i][j] = M[i-1][j]$ and $P[i][j] = P[i-1][j]$. For example, consider the case corresponding to the 16-th row, i.e., the last sub-pattern $\rho(7,8)$, and the maximum length, i.e., 8 in Fig. 5 (highlighted in green). If $\rho(7,8)$ is merged with $P[16][6]$, i.e., $1^3 0^0 1^2$, we get $1^3 0^3 10$ which satisfies the $r_{min} = 0.5$. This merging gives an advantage value of $M[16][6] + D[7][8] = 3.76$ which is more than the maximum advantage value computed (i.e., $M[15][8]$) before considering $\rho(7,8)$ for 8 length patterns (Fig. 9). Therefore, we populate $M[16][8]$ with 3.76 and $P[16][8]$ with $1^3 0^3 10$ (Fig. 9).

	1	2	...	8
1: ρ(2,3)	0	0	...	0.02
⋮	...	...	...	
15: ρ(5,8)	0	0	...	3.7
16: ρ(7,8)	0	0	...	Max(___, D[7][8] + M[16][7-1])

Figure 9: DP case 3iii

We summarize the recursive DP formulation in Eq. 10. The last column of the P matrix, i.e., $P[i][t] \forall i \in [1,p]$ gives the list of attack-resilient control execution patterns of length t, i.e. $\{0,1\}^t$, ranked (from least beneficial to most beneficial) with respect to the advantage values. Thus, by construction, the most attack-resilient t length optimal control execution pattern is stored in $P[p][t]$. We can see in the example of Fig. 5 that $M[16][8]$ has the maximum advantage value with the corresponding pattern stored in $P[16][8]$. The time complexity of this DP-based solution method is $O(pt)$.

$$M[i][j] = \begin{cases} 0 & \text{if } i = 1 \text{ and } j < end0(i) \\ D[i][j] & \text{if } i = 1,\ j \geq end0(i) \wedge rate(i) \geq r_{min} \\ 0 & \text{if } i = 1,\ j \geq end0(i) \wedge rate(i) < r_{min} \\ M[i-1][j] & \text{if } i > 1 \wedge j < end0(i) \\ M[i-1][j] & \text{if } i > 1 \wedge j \geq end0(i) \wedge \\ & rate(1^{end1(i)}0^{end0(i)-end1(i)}1^{j-end0(i)}) \\ & < r_{min} \\ max\{M[i-1][j], & \text{if } i > 1 \wedge j \geq end0(i) \wedge \\ D[end1(i)][end0(i)]\} & rate(1^{end1(i)}0^{end0(i)-end1(i)}1^{j-end0(i)}) \\ & \geq r_{min} \wedge rate(P[i][end1(i)-1] \circ \\ & 1^{end1(i)}0^{end0(i)-end1(i)}1^{j-end0(i)}) \\ & < r_{min} \\ max\{M[i-1][j], & \text{if } i > 1 \wedge j \geq end0(i) \wedge, \\ M[i][end1(i)-1] & rate(P[i][end1(i)-1] \\ +D[end1(i)][end0(i)]\} & \circ 1^{end1(i)}0^{end0(i)-end1(i)}1^{j-end0(i)}) \\ & \geq r_{min} \end{cases} \tag{10}$$

5 ATTACK LIBRARY SYNTHESIS

To evaluate our proposed methodology, a set of FDI attack vectors is required. Depending on the attacker's trend of attack injection, the victim system needs to be equipped with those control execution skip patterns which are the best at minimizing the effect of this attacker's actions on the system. For this, we need to model an attacker whose attack strategy we have to learn online and deploy patterns accordingly. *In this work, we consider an attack model, where an attacker injects false data into actuator and sensor signals in consecutive samples to incur maximum damage to the system's performance in minimum time while being stealthy.* For a given CPS, the notion of minimum-length and stealthy false data injection attack is presented in the following definition.

Definition 4 (**Minimum length Stealthy False Data Injection attack**). *A t length false data injection attack vector $\mathbf{a}_t$ is stealthy (Def. 1) and of minimum length, if it can stealthily steer the system trajectory beyond the safety envelope X_s while no attack vector of smaller length can make it possible, i.e., $x_t^a \notin X_s$ and $f(r_k^a) < Th\ \forall k \in [1,t]$ but $x_k^a \in X_s\ \forall k \in [1,t-1]$.* □

We formally present the problem of generating minimum-length stealthy attack vectors as follows.

$$\mathcal{CP}: \exists\, \mathbf{a}[1], \mathbf{a}[2], \cdots, \mathbf{a}[t] \tag{11}$$

$$\text{s.t.} \quad x_0^a = x;\ \hat{x}_0^a = x \text{ where } x \in X_s \tag{12}$$

$$u_{i-1}^a = -K\hat{x}_{i-1}^a;\ \tilde{u}_{i-1}^a = u_{i-1}^a + a_i^u\ \forall i \in [1, t] \tag{13}$$

$$x_i^a = Ax_{i-1}^a + B\tilde{u}_{i-1}^a;\ y_i^a = Cx_i^a + a_i^y\ \forall i \in [1,\ t] \tag{14}$$

$$r_{i-1}^a = y_i^a - C(A\hat{x}_{i-1}^a + Bu_{i-1}^a); \hat{x}_i^a = A\hat{x}_{i-1}^a + Bu_{i-1}^a + Lr_{i-1}^a \forall i \in [1, t] \tag{15}$$

$$f(r_{i-1}^a) < Th;\ |y_i^a|, |a_i^y| < \mathcal{Y};\ |u_i^a|, |\tilde{u}_i^a|, |a_i^u| < \mathcal{U}\ \forall i \in [1,\ t] \tag{16}$$

$$x_i^a \in X_s \forall i \in [1,\ t-1];\ x_t^a \in X_s \tag{17}$$

The above constraint solving problem $\mathcal{CP}$ returns an attack vector $\mathbf{a}_t = [\mathbf{a}[1], \mathbf{a}[2], \cdots, \mathbf{a}[t]]$ (11) of length t satisfying all the constraints in (12)-(17) for an initial value of the state (12). The constraints (13)-(15) follow the system progression under attack (Eq. 2). The stealthiness of $\mathbf{a}_t$ is ensured by the constraint $f(r_{i-1}^a) < Th$ in 16. The other constraints in (16) guarantee that attacks on the sensor and actuation signals as well as the falsified measurement and actuation signals are within their respective ranges. To make sure the attack vector $\mathbf{a}_t$ is of minimum length (Def. 4), we keep the safety constraints in (17). Initially, we solve this problem using some constraint solver with the value of $t = 1$. If $\mathcal{CP}$ returns no solution, we keep on incrementing the value of t by one until the minimum length attack vector is returned. Varying the initial state value x in (12), we can generate a library of attack vectors $\mathcal{A}_{lib}$ such that $\forall \mathbf{a} \in \mathcal{A}_{lib}, \mathbf{a} = \begin{bmatrix} a_1^u & a_2^u & \cdots & a_l^u \\ a_1^y & a_2^y & \cdots & a_l^y \end{bmatrix}, l \geq 1$.

6 FDI ATTACK PREDICTION

In this section, we describe the design of a lightweight predictor that forecasts the system behavior under attack. The predictor is designed to observe a time series data for a fixed-length time window and predict a sequence of measurements under attack for a future time window. As seen in Eq. 2, the observed falsified system measurements from the controller side are y^a. Let $\tilde{y}^a = Cx^a = y^a - a^y$ denote system output without the sensor attacks (a^y) at current iteration. At any time instant t_0, observing y^a and r^a from time $t_0 - d_{past}$ to t_0, the predictor predicts the most probable values of $\tilde{y}^a$ and r^a for a future time window $[t_0 + 1, t_0 + d_{future} + 1]$. In this case, the observation window length is d_{past} and the prediction window length is d_{future}.

The predictor is designed with 4 layers of neural networks consisting of 2 layers of *gated recurrent units* (GRU) [2] followed by 2 layers of fully connected layers. We choose GRU for this *multivariate time series forecasting* instead of recurrent neural network (RNN) or long-short-term memory (LSTM)-based method *(i)* to eliminate the short-term memory issue of RNN [5] and *(ii)* to keep the implementation lightweight for embedded platforms (something not possible with LSTM). The following equation expresses this operation: $\begin{bmatrix} \tilde{y}_{t_0+1}^a & \cdots & \tilde{y}_{t_0+d_{future}+1}^a \\ r_{t_0+1}^a & \cdots & r_{t_0+d_{future}+1}^a \end{bmatrix} = predictor\Big(\begin{bmatrix} y_{t_0-d_{past}}^a & \cdots & y_{t_0}^a \\ r_{t_0-d_{past}}^a & \cdots & r_{t_0}^a \end{bmatrix}\Big)$. During offline training, we simulate the victim system model in presence of the attack vectors from the library $\mathcal{A}_{lib}$ of attack vectors synthesized in Sec. 5 to generate such under-attack system traces (following Eq. 2). Here, $\Big\langle \big\{[\tilde{y}^a\ r^a]^T\big\}_{d_{future}}, \big\{[y^a\ r^a]^T\big\}_{d_{past}} \Big\rangle$ are the $\langle label, data \rangle$ pairs that are used to train and test the predictor offline (see *Predictor (GRU)* block in Fig. 1). Using the predicted future values of $[\tilde{y}^a\ r^a]^T$, we calculate the most probable future values for the sensor and actuator attacks a_{k+1}^y and a_k^u respectively as follows: $\tilde{\hat{x}}_{k+1} = A\tilde{\hat{x}}_k + Bu_k + L\tilde{r}_{k+1}^a$, $\tilde{r}_{k+1}^a = \tilde{y}_{k+1}^a - C\hat{x}_{k+1}, a_k^u = \tilde{\hat{x}}_{k+1} - \hat{x}_{k+1}$, $a_{k+1}^y = r_{k+1}^a - \tilde{r}_{k+1}^a$. We denote this operation using the function f in Fig. 1. The final output $\big\{[a_{pred}^u\ a_{pred}^y]^T\big\}_{d_{future}} = f\big(\big\{[\tilde{y}^a\ r^a]^T\big\}_{d_{future}}\big)$, where $\{.\}_{d_{future}}$ denotes the values of the variables for d_{future} consecutive sampling iterations and *pred* subscript denotes the predicted data. During the evaluation of the trained predictor, the predicted outputs $\big\{[a_{pred}^u\ a_{pred}^y]^T\big\}_{d_{future}}$ are compared against the generated test data $\big\{[a^u\ a^y]^T\big\}_{d_{future}}$ to derive the output error margin and accuracy (in *Predictor' performance evaluation* block in Fig. 1). At runtime, the trained predictor is deployed on the controller side and infers/predicts *the most probable attack vector for a future time window* (i.e., $\big\{[a_{pred}^u\ a_{pred}^y]^T\big\}_{d_{future}}$) given a past observation $\big\{[y^a\ r^a]^T\big\}_{d_{past}}$.

7 PREDICTED ATTACK CLASSIFICATION

Given a prediction of the most probable FDI for future iterations, we need to deploy an optimal attack-resilient pattern to incapacitate it. For this, we partition the library of attack vectors $\mathcal{A}_{lib}$ synthesized in Sec.5 into a finite number of clusters $C_1, C_2, \cdots, C_m$ such that for any $\mathbf{a}_i, \mathbf{a}_j \in \mathbf{C}_k$ and $\mathbf{a}_i' \in C_k'$, GenPat$(system, \mathbf{a}_i)$ = GenPat$(system, \mathbf{a}_j) \neq$ GenPat$(system, \mathbf{a}_i')$. So basically, the clusters are labeled with control execution skip patterns such that any two attack vectors $\mathbf{a}_i$ and $\mathbf{a}_j$ will be placed in the same cluster $\mathbf{C}_k$ if the optimal attack-resilient control execution patterns returned by our proposed DP-based method (from Sec. 4, denoted using GenPat) for $\mathbf{a}_i$ and $\mathbf{a}_j$ are same. The pattern represents the label of C_k. This way, we generate a labeled dataset C_{lib} required to train the classifier, i.e., $C_{lib} = \big[\big\langle [\mathbf{a}_i, \mathbf{a}_{i+1}, \cdots, \mathbf{a}_{l_i}], C_1 \big\rangle, \big\langle [\mathbf{a}_j, \mathbf{a}_{j+1}, \cdots, \mathbf{a}_{l_j}], C_2 \big\rangle, \cdots, \big\langle [\mathbf{a}_k, \mathbf{a}_{k+1}, \cdots, \mathbf{a}_{l_k}], C_m \big\rangle\big]$. We then build a *multi-layer perceptron (MLP)-based classifier* that can map a predicted attack vector in runtime to a certain cluster and deploy the control execution skip pattern which labels that cluster. MLPs are artificial neural networks that are heavily used as classifiers because of their fast inferencing capability [18]. This constitutes multi-layer fully connected neural networks with desired activation functions ($tanh$ in our case). Using the data set C_{lib}, we train the designed classifier offline. In runtime, once the predictor forecasts the attack vector for a time window, the classifier maps it to an attack vector cluster and decides to deploy the attack-resilient pattern online.

To summarize the overall proposed methodology as shown in Fig. 1, we first **i)** synthesize a library of attack vectors (with different initial points in the system state-space) using which the data set for training and evaluating the predictor and classifier is obtained. **ii)** A GRU-based predictor is designed and trained to guess the most probable attack vector from past data. **iii)** A DP-based attack-resilient control skip pattern synthesis method is designed to optimally minimize the effects of the generated attack vectors. **iv)** The library of attack vectors is partitioned into clusters such that for all attack

vectors in a cluster, there exists one optimal attack-resilient control skip pattern to label that cluster. **v)** An MLP-based classifier is trained to map a predicted attack vector to a suitable cluster to decide which control execution pattern to deploy in order to optimally reduce the predicted attack's effect on the system. After evaluation, the trained predictor and the classifier along with the cluster information are deployed on CPS platforms for resilient control execution at runtime.

Systems	Order	r_{min}	Initial State	Lenth of Attack	Control Skip Patterns	Advantages
Trajectory Tracking Control [1]	2	0.51	[0.65; 0.78]	10	$10^5 1^4$	15.54
					$10^4 1^5$	12.12
					$10^2 1^7$	5.72
ESP [1]	2	0.45	[-1.96; -3.93]	3	101	2.62
Fuel Injection [17]	3	0.5	[-0.08; -0.53; -1.77]	5	$101^2 0$	6.05
					$1^3 0^2$	4.39
					$10^2 1^2$	2.25
					101^3	1.69
Suspension Control [12]	4	0.52	[-1.64; 21.89; 47.19; 71.12]	7	$1010^2 1^2$	936.84
					10101^3	878.63
					$1^2 0^3 1^2$	719.26
					$10^3 1^3$	660.99
Four-Car Platoon [13]	8	0.5	[-0.55; 1.47; -5.29; -0.26; -1.84; -1.39; 4.47; 3.75]';	26	$1^5 0^{13} 1^8$	280.29
					$1^4 0^{13} 1^9$	254.08
					$1^3 0^{13} 1^{10}$	230.37
					$1^2 0^{13} 1^{11}$	208.14

Table 1: Resilient Control Sequences for Automotive Benchmarks

8 EXPERIMENTAL RESULTS

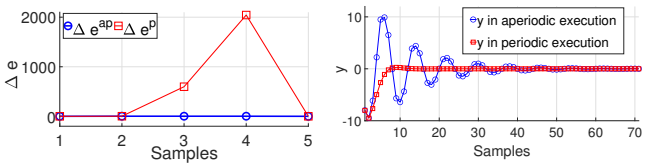
To evaluate our proposed methodology, we consider several safety-critical CPS benchmarks of different dimensions from the automotive domain. For each benchmark, we consider the control performance w.r.t. the settling time, i.e., the duration within which system output reaches and stays within 2% of its reference. The 1st, 2nd, and 3rd columns of Tab. 1 respectively contain the benchmarks' names (along with corresponding references), their dimensions, and minimum control execution rate to maintain the desired performance criteria. The proposed methodology is implemented using Matlab and Tensorflow, Keras, Scikit-learn libraries in Python.

Systems	Cluster Labels
Trajectory Tracking Control	$1^4 0$, $10^2 1^2$, $1^3 1^2$, $10^3 1^3$, $1^4 0^4$, $1^2 0^4 1^2$, $1^5 0^3$, $10^4 1^3$, $1^2 0^4 1^3$, $10^4 1^4$, $10^5 1^4$, $10^6 1^6$, $10^6 1^5$
ESP	101
Fuel Injection	$1^2 0$, 1010, $101^2 0$
Suspension Control	101, $10^2 1$, $1010^2 1^2$
Four-Car Platoon	10, $1^5 0^{13} 1^8$, $10^5 1^5$, $1^6 0^{12} 1^7$, $1^5 0^{12} 1^7$, $1^4 0^{11} 1^8$, $1^5 0^{12} 1^8$, $1^4 0^{11} 1^7$

Table 2: $\mathcal{A}_{lib}$ Clusters

Evaluation of DP-based pattern synthesis method: In Tab. 1, we specify attack-resilient control execution sequences synthesized for each of the benchmarks in order to verify the scalability of our approach. We have implemented the constraint solving problem in Eqs. 11-17 using the popular optimization problem formulation tool YALMIP equipped with Gurobi solver (on an 8-core 7-th gen intel i7 CPU with 16 GB of RAM) to generate a minimum length attack vector for a single initial point (4th column) inside the safety polytope X_s for each benchmark. We run the DP-based solution to synthesize the patterns exhibiting optimal resilience against the synthesized attack vector and satisfying the minimum execution rate r_{min} for performance (3rd column). The length of the patterns is considered the same as the minimum-length attack (5th column). The 6th column provides the list of synthesized control execution patterns ranked in the descending order (last column of P matrix in DP formulation in Sec. 4) w.r.t. their advantage values (last column of M matrix in Eq. 10). It is evident from Tab. 1 that our proposed pattern synthesis method can scale even for higher dimension systems. We sampled 1000 random initial state values from safety region X_s of each benchmark to generate minimum-length attack vectors for each of the initial points and built $\mathcal{A}_{lib}$. $\mathcal{A}_{lib}$ is subsequently used to create the predictor data set. Also, we partition this library into the labeled clusters for MLP classifier training (Sec. 7). The clusters of each benchmark used by us are given in Tab. 2.

In Fig. 10, we demonstrate the resilience and performance of one such synthesized control execution skip pattern for a suspension control system [12]. We consider a predicted attack vector of length 4 mapped to the cluster 1001 of the suspension control system. The blue plot with a circle marker denotes $\triangle e^{ap}$ while following the control execution pattern 1001 and the red plot with a square marker denotes $\triangle e^{p}$ while following periodic control execution. As we can clearly observe in Fig. 10a, under the 4-length attack $\triangle e^{ap}$ is significantly less than $\triangle e^{p}$. Fig. 10b showcases the performance of

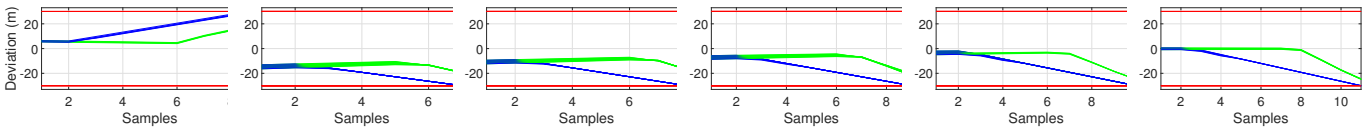


(a) Comparison Between Δe^{p} and Δe^{ap} (b) Periodic & Aperiodic y under no FDI

Figure 10: Effect of Derived Aperiodic Control Execution Sequence $(1001)^{\omega}$ on Suspension Control System under FDI

the system under the aperiodic control sequence 1001 (in the blue and circled plot) and the periodic control execution (in the red and squared plot) without any FDI attack. As per the design criteria, the system must settle within 3 seconds. We can see system output (position of the car in meters) under the periodic control execution settles much more quickly compared to the aperiodic execution. However, since the aperiodic control sequences synthesized using our framework always follows the r_{min}, even under the aperiodic execution, the system output settles within 2.4 seconds (i.e., 60 sampling periods each of 0.04 sec). This successfully validates that the attack-resilient control sequences generated using our framework preserve system performance while turning out to be beneficial in terms of reducing the damage caused by an FDI.

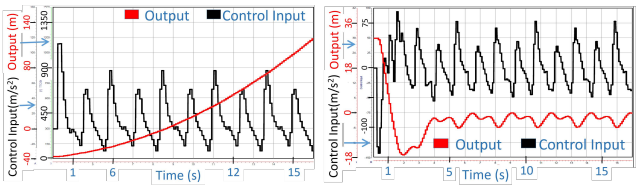
Evaluation of the deployed framework: During the offline training phase, the predictor and the classifier are trained to predict an FDI attacker's intent and deploy patterns to incapacitate the attack effort. The trained models report a mean square error (MSE) of $\sim$ 10%. Thereafter, both are deployed in a loop with various CPS examples for experimental evaluation. Both are observed to infer within the observation window in runtime. For examining the robustness of the proposed attack-resilient pattern synthesis algorithm, we consider the trajectory tracking control (TTC) system. From the predictor test data, we extract a set of attack vectors. For each of them, we add noise with zero mean and variance same as the MSE margin (10%) as mentioned earlier. This creates the test vector library $\in \mathcal{A}_{rand}$ for our evaluation. Next, we run the system



(a) $10^4 1^3$ (b) $10^3 1^2$ (c) $10^4 1^3$ (d) $10^4 1^4$ (e) $10^2 0^4 1^3$ (f) $10^5 1^4$

Figure 11: Robustness analysis of the proposed attack-resilient pattern synthesis method (red: safety boundary, blue: periodic, green: aperiodic)

in a loop with predictor+classifier while injecting attacks sampled from $\in \mathcal{A}_{rand}$. The effect of various execution patterns deployed autonomously by our methodology at runtime for the TTC system can be seen in different plots of Fig. 11a-f. Each plot shows the effect of attack vectors mapped to a single cluster. For all the attack vectors in $\mathcal{A}_{rand}$ when various controller execution patterns are followed, the outputs of TTC (i.e., the deviation from the desired separation among vehicles) are within the green regions in Fig. 11a-f. Similarly, the outputs of TTC for the same attack vectors under periodic execution of the controller are within the blue regions in Fig. 11a-f. The boundaries of the safety region $[-30m, 30m]$ for deviation are indicated by the red lines. The system is safe as long as the output is within those red boundaries. We can see in each of the plots in Fig. 11 that the attack vectors in $\mathcal{A}_{rand}$ take the system beyond the safety boundary within $\sim$ 10 samples during periodic control execution. However, the system output is within the safety region when the control skip pattern decided by our methodology is followed. This provides a larger time window for countermeasure deployment, thus increasing the resilience as claimed in this work.



(a) TTC under minimum-length FDI and periodic execution **(b) TTC under minimum-length FDI with Optimal control skipping pattern**

Figure 12: Effect of optimal attack-resilient control skipping sequence on trajectory tracking control system under FDI attack

Realization on hardware-in-loop (HIL) setup: We demonstrate the real-time applicability of the proposed method using a HIL setup for the TTC system. The controller, estimator, attack-predictor, and attack classifier are implemented on an NVIDIA Jetson Nano board which is equipped with a Quad-core ARM A57 CPU and NVIDIA Maxwell architecture GPU with 128 NVIDIA CUDA cores. The plant is implemented on ETAS LabCar HIL real-time simulator. The controller and the plant communicate via the CAN (Controller Area Network) bus using the MCP2515 CAN controller. We consider the TTC system is under stealthy FDI attack. Since in this case $\mathcal{A}_{lib}$ consists of the attack vectors that have a length of 13 (as explained in Def. 4), we consider a prediction horizon of 13 samples. The first predicted attack vector is mapped to the cluster $10^6 1^6$. The next attack vector, predicted based on the observations from the first prediction window, is mapped to the cluster $10^6 1^5$. First, we apply the predicted stealthy FDI attacks on the TTC system when the controller operates periodically. The output of the TTC system is the deviation from the reference trajectory whose safety range is $[-30m, 30m]$. Fig. 12a shows that the system output gradually becomes unsafe. Next, we consider the controller execution is following the patterns $10^6 1^6$ and $10^6 1^5$ sequentially in repeated loop $((10^6 1^6)(10^6 1^5))^{\omega}$. Fig. 12b shows that the effects of the stealthy FDI attacks have been reduced to a great extent when attack-resilient control execution patterns are applied. Compared to Fig. 12a, the system output in Fig. 12b is still much below the safety boundary of the TTC system's output up to an observation of 10s.

9 CONCLUSION

In this work, we provide an end-to-end methodology for synthesizing attack-resilient control execution patterns concerning a wide range of CPSs. As can be seen, in all cases, the effect of the attack is suitably mitigated by the deployed execution pattern as suggested by our method. As a future extension, we plan to employ this basic strategy against different attack types like *schedule-based, bus-off attack* [6] etc.

REFERENCES

[1] Sunandan Adhikary et al. 2020. Skip to secure: Securing cyber-physical control loops with intentionally skipped executions. In *Joint Workshop on CPS&IoT Security and Privacy*.
[2] KyungHyun Cho et al. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR* abs/1409.1259 (2014).
[3] Seyed Mehran Dibaji et al. 2019. A systems and control perspective of CPS security. *Annual reviews in control* 47 (2019), 394–411.
[4] Sumana Ghosh et al. 2017. A structured methodology for pattern based adaptive scheduling in embedded control. *TECS* 16, 5s (2017).
[5] Sepp Hochreiter et al. 1997. Long short-term memory. *Neural computation* 9, 8 (1997).
[6] S. Hounsinou et al. 2021. Vulnerability of controller area network to schedule-based attacks. In *RTSS*. IEEE.
[7] Ipsita Koley et al. 2020. Formal synthesis of monitoring and detection systems for secure cps implementations. In *DATE*. IEEE.
[8] Vuk Lesi et al. 2017. Security-aware scheduling of embedded control tasks. *TECS* 16, 5s (2017).
[9] Rupak Majumdar et al. 2011. Performance-aware scheduler synthesis for control systems. In *EMSOFT*. IEEE.
[10] Yilin Mo et al. 2010. False data injection attacks in control systems. In *Preprints of the 1st workshop on Secure Control Systems*.
[11] Arslan Munir et al. 2018. Design and analysis of secure and dependable automotive CPS: A steer-by-wire case study. *IEEE TDSC* (2018).
[12] Debayan Roy et al. 2016. Multi-objective co-optimization of FlexRay-based distributed control systems. In *RTAS*. IEEE.
[13] Bastian Schürmann et al. 2017. Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems. In *ACC*. IEEE.
[14] Yasser Shoukry et al. 2013. Non-invasive spoofing attacks for anti-lock braking systems. In *CHES*. Springer.
[15] Damoon Soudbakhsh et al. 2013. Co-design of control and platform with dropped signals. In *ICCPS*.
[16] Andre Teixeira et al. 2015. Secure control systems: A quantitative risk management approach. *IEEE Control Systems Magazine* 35, 1 (2015), 24–45.
[17] Chee Tan Wei et al. 2009. Modeling and control of an engine fuel injection system. *Faculty of Electrical Engineering, University Teknologi Malaysia* 1 (2009).
[18] Yi Zheng et al. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International conference on web-age information management*. Springer, 298–310.

A PROOF OF THEOREM 1

THEOREM 1: For a plant-controller closed-loop system under FDI attack (Eq. 2), control execution skips for consecutive l sampling instances after k periodic control executions will be effective in enhancing the system's resilience when the following criterion is true:
$|| \sum_{i=0}^{l-1} A^i B \triangle a^u_{k+l-1-i} || > || \sum_{i=0}^{l-1} L \triangle r_{k+l-i} ||$
Here, the term $\triangle a^u_{k+l-1-i} = a^u_{k+l-1-i} - a^u_{k-1}, \forall i \in [0, l-1]$ captures the difference between the attacks on control signal on the last l sampling instances out of $(k+l)$ samples between periodic (i.e., none of the $k+l$ samples has been skipped) and aperiodic cases (i.e., last l of the $k+l$ samples has been skipped). □

Proof:
Let $\triangle e^p_{k+l}$ be estimation error deviation after $(k+l)$ periodic executions and $\triangle e^{ap}_{k+l}$ be estimation error deviation after k periodic executions followed by l control execution drops. From Eq. 8 and 9, we get,

$$\triangle e^p_{k+l} = A^l \triangle e_k + \sum_{i=0}^{l-1} A^i (B a^u_{k+l-1-i} - L \triangle r_{k+l-i})$$

$$\triangle e^{ap}_{k+l} = A^l \triangle e_k + \sum_{i=0}^{l-1} A^i B a^u_{k-1}$$

The control execution skips for consecutive l sampling instances after periodic execution of consecutive k sampling instances will enhance the system's resilience against FDI attacks if $||\triangle e^p_{k+l}|| > ||\triangle e^{ap}_{k+l}||$. Now,

$$||\triangle e^p_{k+l}|| > ||\triangle e^{ap}_{k+l}||$$
$$\implies ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i (B a^u_{k+l-1-i} - L \triangle r_{k+l-i})|| > ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i B a^u_{k-1}||$$
$$\implies ||A^l \triangle e_k|| + || \sum_{i=0}^{l-1} A^i (B a^u_{k+l-1-i} - L \triangle r_{k+l-i})|| > ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i B a^u_{k-1}||$$
$$\implies ||A^l \triangle e_k|| - ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i B a^u_{k-1}|| > -|| \sum_{i=0}^{l-1} A^i (B a^u_{k+l-1-i} - L \triangle r_{k+l-i})||$$
$$\implies || - \sum_{i=0}^{l-1} A^i B a^u_{k-1}|| > || \sum_{i=0}^{l-1} A^i (L \triangle r_{k+l-i} - B a^u_{k+l-1-i})||$$
$$\implies || - \sum_{i=0}^{l-1} A^i B a^u_{k-1}|| > || \sum_{i=0}^{l-1} A^i L \triangle r_{k+l-i}|| - || \sum_{i=0}^{l-1} A^i B a^u_{k+l-1-i}||$$
$$\implies || \sum_{i=0}^{l-1} A^i B a^u_{k+l-1-i}|| - || \sum_{i=0}^{l-1} A^i B a^u_{k-1}|| > || \sum_{i=0}^{l-1} A^i L \triangle r_{k+l-i}||$$
$$\implies || \sum_{i=0}^{l-1} A^i B a^u_{k+l-1-i} - \sum_{i=0}^{l-1} A^i B a^u_{k-1}|| > || \sum_{i=0}^{l-1} A^i L \triangle r_{k+l-i}||$$
$$\implies || \sum_{i=0}^{l-1} A^i B \triangle a^u_{k+l-1-i}|| > || \sum_{i=0}^{l-1} A^i L \triangle r_{k+l-i}||$$

□