Exploring The Resilience of Control Execution Skips against False Data Injection Attacks

Ipsita Koley
Indian Institute of Technology,
Kharagpur
India
ipsitakoley@iitkgp.ac.in

Sunandan Adhikary
Indian Institute of Technology,
Kharagpur
India
mesunandan@kgpian.iitkgp.ac.in

Soumyajit Dey
Indian Institute of Technology,
Kharagpur
India
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. We also propose a methodology to synthesize such optimal control execution patterns. 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.

KEYWORDS

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

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. Numerous such attacks on safety-critical CPSs have been reported in the past, for example, Stuxnet[7], Maroochy water breach attack[17], Black energy attack[13], attacks in automotive domain[2, 4], etc.

In this work, we consider a type of attack called false data injection (FDI). Networked control CPSs are designed as a closed-loop where the controller 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. Sometimes, all the states of the plant can not be measured and an observer (like Kalman filter) is used at the controller end to estimate the states of the plant. As the closed loop communication happens over a network, an external or internal attacker can malign the sensor measurements and/or the control signals physically or through the network. In such FDI attacks, the controller and the plant do not receive actual data, leading to instability or performance loss.

Till date the best defence mechanism against FDI attacks is to use cryptographic methods. However, the major hindrance to their application is the computation and communication loads incurred by these methods [8, 12]. An alternate solution that can be found in literature is to use residue-based light-weight attack detection methods [6, 11, 19] interleaved with traditional cryptographic techniques instead of using the latter continuously [1, 5]. A typical FDI attack can not maintain its stealthiness while the transmitted data is secured with cryptographic methods. On the other hand, the statistical nature of the residue-based detectors take some time to detect an FDI attack with higher probability. A smart attacker can intelligently craft the worst-possible FDI attack that can bypass the residue-based attack detectors when cryptographic methods are not active [1, 19]. Therefore, irrespective of what security enforcement is in place (whether the combination of cryptographic method and residue-based detection or continuous use of cryptographic method), the FDI attacks can significantly affect the system's performance. Thus, the question that rises in this context is *how to make the system more resilient against FDI attacks?* In this paper, our objective is to address this question.

The authors of [1] were the first to explore the skipping of some control executions to enhance system's resilience against stealthy FDI attacks while it is not detected. Pattern-based execution of controller where some of the instances of control executions are dropped or skipped was initially studied to accommodate multiple tasks on resource constrained embedded platforms [3, 9, 18]. Skipping or dropping a control execution at a certain sampling instance means no new control input will be computed or communicated at that instance. So, the processor and the communication channel between the plant and controller both will also be free during that sampling instance. It is evident that the FDIs during skips are rendered ineffective. Thus, this can restrain the effectiveness of attacker's effort by enhancing system's resilience. But this skipping of control execution may degrade the performance of the control system. To address this, a *minimum rate of control execution* is required to maintain the desired control performance [3]. Constraining this minimum rate of control execution, the authors of [1] developed a formal methodology-based approach to synthesize control execution-skipping sequences to enhance system security and safety. However, the major limitations of their approach is that, it does not relate the position of control skips with the dynamics of the system under attack, and the SMT-based attack and control execution pattern synthesis might not scale for systems with larger

dimension. Also, if the position of control execution skips are not chosen wisely, the performance of the controller may degrade.

Similar to [1], we also utilize the skipping of control executions to enhance the resilience of the system under attack which may seem counter-intuitive as control performance may degrade due to execution skips. However, the safety-critical CPSs mostly operate at higher sampling rate. The desired performance of such fast systems can be maintained if we can judiciously choose when to skip the control executions. Such control execution skips in turn will ignore that attacks injected at those sampling instances.

In this work, we theoretically analyse and establish analytical conditions under which execution skips will surely be beneficial in terms of enhancing the system's resilience against FDI attacks. We present a methodology to synthesize the most optimal attack-resilient control execution sequence that also ensures the desired performance. We provide an automated CAD tool-chain to generate such control execution sequences given any CPS. To this end, we now summarize the contributions of this work as follows.

(1) Given the specifications of a safety-critical CPS and its initial region, we formulate a constraint solving problem to generate the optimal or worst case FDI attack sequence that consumes minimum time to make the system unsafe.

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

(3) Utilizing the conditions established in 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 optimal FDI attacks generated in contribution 1 .

(4) We provide an automated CAD tool that takes as input the CPS specification and synthesizes resilient control execution patterns for the same. The scalability of the proposed methodology has been evaluated on well known benchmarks with various dimensions.

Figure 1: Secure CPS architecture

2 BACKGROUND

Secure CPS Model: General architecture of a secure CPS is presented in Fig. 1. The physical process i.e. the plant and the controller work together in a closed loop manner such that the desired

operating criteria 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.

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

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. As security enforcement, we consider a sporadic implementation of some cryptographic method along with a residue-based detector as demonstrated in Fig. 1. 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.

Consider an FDI attack, where the attacker injects false data a_k^y and a_k^u (Fig. 1) to the sensor measurement and control signal respectively when the cryptographic method is not active (see Fig. 1). In such scenario, the system dynamical equation becomes,

$$x_{k+1}^a = Ax_k^a + B\tilde{u}_k^a + w_k; \; y_k^a = Cx_k^a + v_k + a_k^y$$
$$\hat{y}_{k+1}^a = C(A\hat{x}_k^a + Bu_k^a); \; r_{k+1}^a = y_{k+1}^a - \hat{y}_{k+1}^a$$
$$\hat{x}_{k+1}^a = A\hat{x}_k^a + Bu_k^a + Lr_{k+1}^a; u_k^a = -K\hat{x}_k^a; \tilde{u}_k^a = u_k^a + a_k^u; e_k^a = x_k^a - \hat{x}_k^a; \quad (2)$$

Here, $x_k^a, \hat{x}_k^a, y_k^a, r_k^a, u_k^a, \tilde{u}_k^a$, and e_k^a 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_k^a is the control input computed at k-th sampling instance on which the effect previous attacks i.e. a_i^u and a_i^y for $1 \le i \le k-1$ persist. u_k^a added with attack on actuator a_k^u at k-th sample produces $\tilde{u}_k^a$ i.e. the forged control signal at k-th sample. Note that even though we discussed about intrusion through network, physical level sensor data tampering [16] can also happen. The above attack model is generic to all kind of such falsification attacks. We denote an attack vector at k-th sampling instance as $\mathcal{A}[k] = [a_k^u, a_k^y]^T$. If the attacker continues the false data injection for l sampling iterations, then the l length attack vector is expressed as follows $\mathcal{A}_l = [\mathcal{A}[1] \cdots \mathcal{A}[l]] = \begin{bmatrix} a_1^u & \cdots & a_l^u \\ a_1^y & \cdots & a_l^y \end{bmatrix}$. 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 limit. On the other hand, it modifies the sensor measurements with a sequence of a^y's such that it can hide itself from the residue based detector. We define such *stealthy* FDI attack vector as follows.

DEFINITION 1 (**Stealthy false data injection attack**). *An attack vector $\mathcal{A}_l = [\mathcal{A}[1] \cdots \mathcal{A}[l]] == \begin{bmatrix} a_1^u & \cdots & a_l^u \\ a_1^y & \cdots & a_l^y \end{bmatrix}$ of length l is said to be stealthy if $f(r_k^a) < Th \; \forall k \in [1, l]$ where r_k^a is the residue generated due the attack vector $\mathcal{A}_l$ at k-th sampling instance.* □

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 are as follows.

(1) The sensor measurements y_k are not communicated to the controller unit.
(2) A fresh control input u_k is not calculated and communicated to the plant. The plant updates its states simply using the previous control input
(3) Detection unit will also not operate.

We consider that such skips in control execution shall be regular leading to a pattern in lines of [3]. This is naturally required for deterministic system design and deployment. We provide a formal definition of control execution skipping pattern.

Figure 2: 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 with period t, 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. [3].* □

Symbolically, a pattern can be denoted as $(1^k 0^l)^t$ where $k, t > 0$ and $l \geq 0$. 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. Consider that there is a skip in control execution at $(k+1)$-th sample as demonstrated in Fig. 2. Then according to skip properties, $u_k = u_{k-1}, a_k^u = a_{k-1}^u$, 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.

$$x_{k+1}^a = Ax_k^a + B(u_{k-1}^a + a_{k-1}^a) + w_k; \; \hat{x}_{k+1}^a = A\hat{x}_k^a + Bu_{k-1}^a;$$
$$x_{k+1} = Ax_k + Bu_{k-1} + w_k; \; \hat{x}_{k+1} = A\hat{x}_k + Bu_{k-1};$$
$$e_{k+1}^a = Ae_k^a + Ba_{k-1}^u + w_k; \; e_{k+1} = Ae_k + w_k; \qquad (3)$$

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 2% band 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} [3] 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 PROBLEM FORMULATION

A Motivating example: In this section, we demonstrate how occasional skips in control execution improve the system's resilience against FDI attacks with help of a motivating example (Fig. 3). We take an example of 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. Attacker adds false data to the measurement data(i.e. deviation) and the control signal(i.e. acceleration). In Fig. 3a, 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 3: 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 an 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. 3a). 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. 3b-3d. We can observe that when the attacker injects optimal attack values (Fig. 3a), there is significant deviation in system's state progression (Fig. 3b and 3c) when periodic control execution takes place. However, due to introduction of drops in the control execution, FDIs rendered ineffective on system's state progression (Fig. 3b and 3c) in case of aperiodic control execution. This is because the FDIs in periodic execution increases the estimation error e^a considerably (Fig. 3d). 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 contains e^a within much lower range (Fig. 3d). Thus, we can see the aperiodic control execution following the pattern ρ(Fig. 3a) enhances TTC's resilience against the FDI attack given in Fig. 3a.

Keeping in mind the stealthy FDI attack and its effect on system's state progression, how to generate the pattern to weaken the attack effect most effectively, is discussed in Sec. 4. In the following section, we provide a quantitative analysis behind such motivation of this work.

Analysing effect of execution skips : During execution skips, the attacks on sensor data and control signal 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 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 the 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 system's response in presence and absence of FDI attacks in case of periodic control execution, we introduce the following two terms:

$$\triangle e_k = e_k^a - e_k = (A - LCA)\triangle e_k + (B - LCB)a_k^u - La_{k+1}^y$$
$$= A\triangle e_k + Ba_k^u - L\triangle r_{k+1} = \sum_{i=0}^{k-1} A^i(Ba_{k-1-i}^u - L\triangle r_{k-i}) \ (\triangle e_0 = a_0^y = 0) \ (4)$$
$$\triangle r_k = r_k^a - r_k = CA\triangle e_k + CBa_k^u + a_{k+1}^y \tag{5}$$

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,

$$\triangle e_{k+1} = e_{k+1}^a - e_{k+1} = (Ae_k^a + Ba_{k-1}^u + w_k) - (Ae_k + w_k);$$
$$= A\triangle e_k + Ba_{k-1}^u \tag{6}$$
$$\triangle r_{k+1} = 0 \tag{7}$$

Now, to show whether the execution skips actually enhance the resilience of the system against FDI attacks, we compare two parameters: i) $\triangle e_{k+l}^p$ i.e. estimation error deviation after $(k+l)$ periodic executions and ii) $\triangle e_{k+l}^{ap}$ i.e. estimation error deviation after k periodic executions followed by l control execution drops. This is demonstrated in Fig. 2. 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_k^a is not transmitted, the control input u_k^a is neither computed and transmitted, and the state progresses with last received control signal $\tilde{u}_k^a$ (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_{k+l+1}^a is transmitted to the plant, the control input u_{k+l+1}^a is again computed using y_{k+l+1}^a (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 $\triangle e_{k+l}^p$ and $\triangle e_{k+l}^{ap}$ as,

$$\triangle e_{k+l}^p = A\triangle e_{k+l-1} + Ba_{k+l-1}^u - L\triangle r_{k+l}$$
$$= A[A\triangle e_{k+l-2} + Ba_{k+l-2}^u - L\triangle r_{k+l-1}] + Ba_{k+l-1}^u - L\triangle r_{k+l}$$
$$= A^2\triangle e_{k+l-2} + (ABa_{k+l-2}^u + Ba_{k+l-1}^u) - (AL\triangle r_{k+l-1} + L\triangle r_{k+l})$$
$$= \cdots$$
$$= A^l\triangle e_k + (A^{l-1}Ba_k^u + A^{l-2}Ba_{k+1}^u + \cdots + Ba_{k+l-1}^u)$$
$$- (A^{l-1}L\triangle r_{k+1} + A^{l-2}L\triangle r_{k+2} + \cdots + L\triangle r_{k+l})$$
$$= A^l\triangle e_k + \sum_{i=0}^{l-1} A^i(Ba_{k+l-1-i}^u - L\triangle r_{k+l-i}) \tag{8}$$

$$\triangle e_{k+l}^{ap} = A\triangle e_{k+l-1} + Ba_{k-1}^u = A[A\triangle e_{k+l-2} + Ba_{k-1}^u] + Ba_{k-1}^u$$
$$= A^2\triangle e_{k+l-2} + (A+I)Ba_{k-1}^u$$
$$= \cdots$$
$$= A^l\triangle e_k + (A^{l-1} + A^{l-2} + \cdots + 1)Ba_{k-1}^u = A^l\triangle e_k + \sum_{i=0}^{l-1} A^iBa_{k-1}^u \tag{9}$$

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 same. And, the second term in Eq. 8 and 9 captures effect of 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 the FDI attacks. Occasional skips in control execution will be useful in enhancing 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 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 system's resilience when the following criteria is true:*
$$|| \sum_{i=0}^{l-1} A^iB\triangle a_{k+l-1-i}^u || > || \sum_{i=0}^{l-1} L\triangle r_{k+l-i}||$$
Here, the term $\triangle a_{k+l-1-i}^u = a_{k+l-1-i}^u - a_{k-1}^u, \forall i \in [0, l-1]$ i.e. 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. l out of the $k+l$ samples has been skipped). □

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

$$||\triangle e_{k+l}^{p}|| > ||\triangle e_{k+l}^{ap}||$$

$$\implies ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i (Ba_{k+l-1-i}^u - L\triangle r_{k+l-i})|| > ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i Ba_{k-1}^u||$$

$$\implies ||A^l \triangle e_k|| + ||\sum_{i=0}^{l-1} A^i (Ba_{k+l-1-i}^u - L\triangle r_{k+l-i})|| > ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i Ba_{k-1}^u||$$

$$\implies ||A^l \triangle e_k|| - ||A^l \triangle e_k + \sum_{i=0}^{l-1} A^i Ba_{k-1}^u|| > -||\sum_{i=0}^{l-1} A^i (Ba_{k+l-1-i}^u - L\triangle r_{k+l-i})||$$

$$\implies ||-\sum_{i=0}^{l-1} A^i Ba_{k-1}^u|| > ||\sum_{i=0}^{l-1} A^i (L\triangle r_{k+l-i} - Ba_{k+l-1-i}^u)||$$

$$\implies ||-\sum_{i=0}^{l-1} A^i Ba_{k-1}^u|| > ||\sum_{i=0}^{l-1} A^i L\triangle r_{k+l-i}|| - ||\sum_{i=0}^{l-1} A^i Ba_{k+l-1-i}^u||$$

$$\implies ||\sum_{i=0}^{l-1} A^i Ba_{k+l-1-i}^u|| - ||\sum_{i=0}^{l-1} A^i Ba_{k-1}^u|| > ||\sum_{i=0}^{l-1} A^i L\triangle r_{k+l-i}||$$

$$\implies ||\sum_{i=0}^{l-1} A^i Ba_{k+l-1-i}^u - \sum_{i=0}^{l-1} A^i Ba_{k-1}^u|| > ||\sum_{i=0}^{l-1} A^i L\triangle r_{k+l-i}||$$

$$\implies ||\sum_{i=0}^{l-1} A^i B\triangle a_{k+l-1-i}^u|| > ||\sum_{i=0}^{l-1} A^i L\triangle r_{k+l-i}|| \qquad \square \qquad (10)$$

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_{k-1}^u gets forwarded through all of the next $(l-1)$ iterations. Whereas, in case of periodic control executions, an attacker can inject different actuation attack values $a_{k+l-1-i}^u$ 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 jeopardise the system safety faster (i.e., $||\sum_{i=0}^{l-1} A^i B\triangle a_{k+l-1-i}^u|| \neq 0$), skipping the control execution at that sampling instance helps make the system more resilient against the injected false data.* $\square$

Formal Problem Statement: Consider a plant with system matrices A, B, and C, controller K, observer L, and its initial region X_0. We now formally define the attack-resilient control execution skipping pattern as follows.
For the given system specifications $\langle A, B, C, K, L, X_0 \rangle$, how we can find attack-resilient control execution skipping patterns $\rho = (1^k 0^l)^t$ where $n, k > 0$, $l \geq 0$ utilizing the relation $||\sum_{i=0}^{l-1} A^i B\triangle a_{k+l-1-i}^u|| > ||\sum_{i=0}^{l-1} A^i L\triangle r_{k+l-i}||$ provided minimum execution rate r_{min} of the controller is maintained?

4 PROPOSED METHODOLOGY

In the last section, we identified analytical conditions, which if satisfied can make skips in control execution to be beneficial in enhancing a system's resilience against stealthy FDI attacks. Now, we present our proposed framework for synthesizing attack-resilient control execution patterns. The outline of the framework is demonstrated in Fig. 4.

Figure 4: Framework for attack-resilient control sequence synthesis

The framework (Fig. 4) requires the followings inputs: i) System specifications matrices A, B, and C, controller gain K, observer gain L, the maximum limit $\mathcal{Y}$ of the sensor measurements, actuation saturation limit $\mathcal{U}$, and the threshold Th of the detector in place, ii) Performance criteria of the controller i.e. minimum execution rate r_{min}, iii) The safety property of the system, defined as a safety polytope $X_s \in \mathbb{R}^n$ which mandates that the system trajectory will always be within X_s, and iv) Initial region of the plant states $X_0 \in \mathbb{R}^n$ from which the system progression initiates. With these inputs, the framework sequentially runs two primary functional modules (Fig. 4). The *first* one synthesizes an attack vector that consumes minimum time to make the system unsafe considering the system may initiate any where from X_0. The *second* one generates attack-resilient control execution sequences based on the synthesized attack vector from the first module using a DP based method. These two functional modules are elaborately discussed in Sec. 4.1 and 4.2 respectively.

4.1 Minimum-length Attack Vector Generation

For a given CPS, the notion of minimum length and stealthy false data injection attack is presented in the following definition.

DEFINITION 3 (**Minimum length Stealthy False Data Injection attack**). *A t length false data injection attack vector $\mathcal{A}_t$ 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 $x_k^a \in X_s\ \forall k \in [1, t-1]$.* $\square$

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

$$\mathcal{CP} :\exists\ \mathcal{A}[1], \mathcal{A}[2], \cdots, \mathcal{A}[t] \quad \forall x \in X_0 \tag{11}$$

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

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

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

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

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

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

The above constraint solving problem $\mathcal{CP}$ returns an attack vector $\mathcal{A}_t = [\mathcal{A}[1], \mathcal{A}[2], \cdots, \mathcal{A}[t]]$ (11) of length t satisfying all the constraints in (12)-(12) for any initial value of the state (11). The constraints (13)-(15) follow the system progression under attack (Eq. 2). The stealthiness of $\mathcal{A}_t$ is ensured by the constraint $f(r_{i-1}^a) < Th$ in 16. The other constraints in (16) guarantee that the attack is stealthy and attacks on sensor and actuation signal as well as the falsified measurement and actuation signals are within their respective ranges. To make sure the attack vector $\mathcal{A}_t$ is of minimum length (Def. 3), we keep the safety constraints in (17). Initially, we

solve this problem using some constraint solver with value of $t = 1$. If $C\mathcal{P}$ returns no solution, we keep on incrementing the value of t by one until the minimum length attack vector is returned.

4.2 Attack-Resilient Control Execution Sequence Synthesis

In this section, we will generate optimal attack-resilient control skip patterns in 2 steps with respect to the minimum length attack vector $\mathcal{A}$ that the constraint solving problem $C\mathcal{P}$ returns in (11). *First,* utilizing the condition presented in Theorem 1, we generate a list of t length *sub-patterns* that are beneficial in enhancing system's resilience against FDI attacks. In *second* step, we formulate a DP based solution method to compute the final i.e. optimal 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 two steps.

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
Ensure: List of favourable sub-patterns $subPatternList$ and their advantage metric D
1: **function** ADVPATSYN($A, B, C, K, L, \mathcal{Y}, \mathcal{U}, Th, X_s, X_0$)
2:     $D[i][i] \leftarrow 0 \; \forall i \in [1, t]$; $subPatternList \leftarrow null$;   ▷ Initialization
3:     $\mathcal{A} \leftarrow$ CALLSOLVER($C\mathcal{P}, A, B, C, K, L, \mathcal{Y}, \mathcal{U}, Th, X_s, X_0$);▷ Solve $C\mathcal{P}$ in (11)
4:     $t \leftarrow length(\mathcal{A})$;
5:     $\Delta r \leftarrow$ RESDIFFGEN($A, B, C, K, L, X_0, \mathcal{A}, t$);   ▷ $\Delta r[i] = r_i^a - r_i$ (Eq. 5)
6:     **for** k=1 to t **do**
7:         **for** l=1 to t-k **do**
8:             $lhs \leftarrow 0$; $rhs \leftarrow 0$; $\rho(k, l) \leftarrow 1^t$;
9:             **for** i=1 to l **do**
10:                 **if** k>1 **then** $lhs \leftarrow lhs + A^{i-1}B(a_{k+l-i}^u - a_{k-1}^u)$;
11:                 **else** $lhs \leftarrow lhs + A^{i-1}Ba_{k+l-i}^u$;   ▷ We assume $a_0^u = 0$
12:                 $rhs \leftarrow rhs + A^{i-1}L\Delta r[k + l - i + 1]$;
13:             **if** $||lhs|| > ||rhs||$ **then**
14:                 $\rho(k, l) \leftarrow 1^k 0^{l-k} 1^{t-l}$;
15:                 **if** $sum(\rho(k, l)) \geq r_{min} \times t$ **then** $D[k][k+l] \leftarrow ||lhs|| - ||rhs||$;
16:                 **else** $\rho(k, l) \leftarrow 1^t$;
17:             $subPatternList \leftarrow subPatternList \cup \rho(k, l)$;
        return $D, subPatternList$;

Algorithm 1: Favourable Sub-pattern Synthesis

4.2.1 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 first k iterations, followed by execution skips till l-th iteration and then $(t - l)$ periodic executions. The control execution following a sub-pattern $\rho(k, l)$ is quantified with an *advantage* value as defined next.

DEFINITION 4 (**Advantage value of a sub-pattern**). *The advantage of control execution that follows the sub-pattern* $\rho(k, l) = 1^k 0^{l-k} 1^{t-l}$ *of length t over periodic control execution of length t is quantified by the value* $||\Delta e^p|| - ||\Delta e^{ap}|| = || \sum_{i=0}^{l-1} A^i B \Delta a_{k+l-1-i}^u || - || \sum_{i=0}^{l-1} A^i L \Delta r_{k+l-i}|| $ *(Theorem 1).*   □

We present a method to synthesize a list $subPatternList$ of 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 inputs to the proposed framework (Fig. 4) are passed to Algo. 1. In line 2, we

initialize the advantage matrix D with 0 and $subPatternList$ as null. We solve the constraint solving problem $C\mathcal{P}$ in (11) to generate minimum length stealthy attack $\mathcal{A}$ (line 3) and store its length t in line 4. By simulating the system's state progression under no attack and under the attack $\mathcal{A}$ in a periodic control execution for t iterations (following the Eq. 1 and 2), we compute Δr_i^a for all $i \in [1, t]$ and store them in the array Δr (line 5). The for loop in line 6 signifies the possible number of consecutive 1's in the sub-pattern $\rho(k, l)$ and the for loop in line 7 signifies the number of consecutive 0's following the consecutive 1's. The LHS and RHS of the criteria $|| \sum_{i=0}^{l-1} A^i B \Delta a_{k+l-1-i}^u || > || \sum_{i=0}^{l-1} A^i L \Delta r_{k+l-i}||$ (Theorem 1) are computed in lines 10-11 and line 12 respectively. In line 13, we check if the LHS is more than the RHS i.e. the difference in estimation error under periodic execution Δe^p is more than that of aperiodic execution Δe^{ap}. If yes, then we generate a new sub-pattern $\rho(k, l)$ by introducing skips from $(k + 1)$ to $(k + l)$ in line 14 for generation of new sub-pattern candidate. Next, in line 15, we update the advantage value matrix D with $||lhs|| - ||rhs||$ (which is nothing but $(||\Delta e^p|| - ||\Delta e^{ap}||)$) 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. Next, we include $\rho(k, l)$ (periodic or aperiodic) into $subPatternList$ (line 17). 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 $O(t^3)$.

4.2.2 Optimal Attack-resilient Pattern Synthesis. In this section, we 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 minimum-length attack vector $\mathcal{A}$ 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 the length of minimum length attack vector $\mathcal{A}$ 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 are $subPatternList$ 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. 4) that can be achieved by considering skips till j-th position in the pattern of length 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 index 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 merging of two patterns i and j using element wise logical AND operation and denote the merged pattern by $i \circ j$. For example, $D[2][3] = 0.02$ implies the advantage value of $\rho(2, 3)$ is 0.02. With help of this example, we now elaborate how to populate M (Eq. 18) and P using DP memoization process.

case 1 ($i = 1$): Consider the 1-st sub-pattern in the list $subPatternList$ to be used (as per definition) to populate first row of M and P. Since we do not have any favourable sub-pattern with skips

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: $\rho(2,3)$	0	0	0.02	0.02	0.02	0.02	0.02	0.02
2: $\rho(2,4)$	0	0	0.02	0.08	0.08	0.08	0.08	0.08
3: $\rho(2,5)$	0	0	0.02	0.08	0.08	0.15	0.15	0.15
4: $\rho(2,6)$	0	0	0.02	0.08	0.08	0.15	0.15	3.7
5: $\rho(3,4)$	0	0	0.02	0.08	0.08	0.15	0.15	3.7
6: $\rho(3,5)$	0	0	0.02	0.08	0.12	0.15	0.15	3.7
7: $\rho(3,6)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
8: $\rho(3,7)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
9: $\rho(4,5)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
10: $\rho(4,6)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
11: $\rho(4,7)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
12: $\rho(4,8)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
13: $\rho(5,6)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
14: $\rho(5,7)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
15: $\rho(5,8)$	0	0	0.02	0.08	0.12	3.68	3.68	3.7
16: $\rho(7,8)$	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: $\rho(2,3)$	11111111	11111111	11011111	11011111	11011111	11011111	11011111	11011111
2: $\rho(2,4)$	11111111	11111111	11011111	11001111	11001111	11001111	11001111	11001111
3: $\rho(2,5)$	11111111	11111111	11011111	11001111	11001111	11000111	11000111	11000111
4: $\rho(2,6)$	11111111	11111111	11011111	11001111	11001111	11000111	11000111	11000011
5: $\rho(3,4)$	11111111	11111111	11011111	11001111	11001111	11000111	11000111	11000011
6: $\rho(3,5)$	11111111	11111111	11011111	11001111	11100111	11100111	11100011	11000011
7: $\rho(3,6)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
8: $\rho(3,7)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
9: $\rho(4,5)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
10: $\rho(4,6)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
11: $\rho(4,7)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
12: $\rho(4,8)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
13: $\rho(5,6)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
14: $\rho(5,7)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
15: $\rho(5,8)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11000011
16: $\rho(7,8)$	11111111	11111111	11011111	11001111	11100111	11100011	11100011	11100010

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

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. This is corresponding 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 that produces positive advantage value (i.e., **[1]**011111 or **[11]**011111). 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. $1^{end1(1)} 0^{end0(1)-end1(1)} 1^{j-end0(1)} = 1^2 0 1^{j-3}$ satisfies the rate criteria. Otherwise, $M[i][j]$ and $P[i][j]$ are assigned 0 and 1^t respectively. Let us again consider the example given 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 $1^2 0^{3-2} 1^{8-3} = 1^2 0 1^5 \; \forall j \geq 3$.

case 2 ($i > 1$ and $j < end0(i)$): Now, let us consider the other sub-patterns in $subPatternList$. There is no prefix up to length $(end0(i) - 1)$ in the i-th sub-pattern $[1^{\mathbf{end1}(i)} 0^{\mathbf{end0}(i) - \mathbf{end1}(i) - 1}]0 1^{t-end0(i)}$ that produces positive advantage value. Therefore, for $i > 1, j < end0(i)$, we can update $M[i][j]$ with $M[i - 1][j]$. 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 that yields a positive advantage value in the current sub-pattern $[\mathbf{110}]01111$. Therefore, we set $M[2][j] = M[1][j]$ 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 $P[1][j]$.

$$
M[i][j] = \begin{cases}
0 \text{ if } i = 1 \text{ and } j < end0(i) \\
D[i][j] \text{ if } i = 1, j \geq end0(i) \wedge rate(i) \geq r_{min} \\
0 \text{ if } i = 1, j \geq end0(i), \text{ and } rate(i) < r_{min} \\
M[i-1][j] \text{ if } i > 1 \wedge j < end0(i) \\
M[i-1][j] \\
\quad \text{if } i > 1 \wedge j \geq end0(i) \wedge \\
\quad rate(1^{end1(i)} 0^{end0(i)-end1(i)} 1^{j-end0(i)}) < r_{min} \\
max\{M[i-1][j], D[end1(i)][end0(i)]\} \\
\quad \text{if } i > 1 \wedge j \geq end0(i) \wedge \\
\quad rate(1^{end1(i)} 0^{end0(i)-end1(i)} 1^{j-end0(i)}) \geq r_{min} \wedge \\
\quad rate(P[i][end1(i)-1] \circ 1^{end1(i)} 0^{end0(i)-end1(i)} 1^{j-end0(i)}) \\
\hfill < r_{min} \\
max\{M[i-1][j], M[i][end1(i)-1] + D[end1(i)][end0(i)]\} \\
\quad \text{if } i > 1 \wedge j \geq end0(i) \wedge, \\
\quad rate(P[i][end1(i)-1] \circ 1^{end1(i)} 0^{end0(i)-end1(i)} 1^{j-end0(i)}) \\
\hfill \geq r_{min}
\end{cases}
$$
(18)

case 3 ($i > 1$ and $j \geq end0(i)$): We divide this case in 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 $P[i][j] = P[i-1][j]$. 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 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 $P[4][6] = P[3][6]$.

(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 same position. Therefore, the candidate patterns which can be merged with i-th sub-pattern $1^{end1(i)} 0^{end0(i)-end1(i)-1} 0 1^{t-end0(i)}$ 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 even the if they are non-overlapping. If the rate condition on the merged pattern is not satisfied, we set $M[i][j] = max\{M[i-1][j], D[end1(i)][end0(i)]\}$ and accordingly 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 $\rho(4, 8) = 1^4 0^{8-4} 1^0 = 11110000$. In $M[12][8]$, we want to have the maximum advantage value that can be achieved by considering skips until last position of 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 mergable with 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 $M[12][8] = max\{M[11][8], D[4][8]\} = M[11][8]$, 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 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. Therefore, we populate $M[16][8]$ with 3.76 and $P[16][8]$ with $1^3 0^3 10$.

The last column of P matrix i.e. $P[i][t] \forall i \in [1, p]$ gives the list of attack-resilient control execution patterns of length 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 $O(pt)$.

5 EXPERIMENTAL RESULTS

For evaluation of our framework, we consider several safety-critical CPS benchmarks. The system descriptions are given as the input to our tool along with their initial region, performance and safety criteria as mentioned in Fig. 4. The framework is built using Matlab and is shared in a public repository [1]. Our experiments are run on an 8-core 7-th gen intel i7 CPU with 16 GB of RAM.

In Tab. 1 we demonstrate attack-resilient control execution sequences synthesized for control systems (provided with corresponding references in the 1st column) with different dimensions (provided in the 2nd column of the table) in order to verify the scalability of our approach. The synthesized patterns satisfy the minimum execution rate r_{min} (3rd column) and their length is considered same as the minimum-length attack (4th column) discovered using Eq. 11-17. The 6th and 7th column provides the list of synthesized control execution patterns ranked in descending order w.r.t. resilience and

Table 1: Resilient Control Sequences for Automotive Benchmarks

Systems	Dime-nsion	r_{min}	Minimum length of Attack	Pattern Synthesis Time (s)	Control Execution Sequences	Advantages
Trajectory Tracking Control [1]	2	0.51	13	0.55s [Total: 21.55s]	$10^6 1^6$	13.58
					$10^5 1^7$	9.86
					$10^4 1^8$	6.14
					$10^3 1^9$	2.05
ESP [1]	2	0.45	3	0.054s [Total: 4.18s]	101	2.62
Fuel Injection [20]	3	0.5	8	0.38s [Total: 13.26s]	$10^3 1^3 0$	7.03
					$10^2 1^3 0^2$	6.49
					$1^5 0^3$	4.22
					$1^4 0^4$	3.86
Suspension Control [14]	4	0.52	4	0.12s [Total: 7.08s]	$1^2 0^2$	2274.73
					$10^2 1$	2096.29
					101^2	434.14
Four-Car Platoon [15]	8	0.5	25	3.56s [Total: 272.62s]	$10^{12} 1^{12}$	4.64
					$10^{11} 1^{13}$	4.01
					$10^{10} 1^{14}$	3.44
					$10^9 1^{15}$	2.92

their advantage values. In the 5-th column we provide the runtime of the pattern synthesis methodology (along with the runtime of the overall methodology in braces). The system descriptions along with safety and performance criteria are taken from [10, 14]. Further, in Figure 6, we demonstrate the resilience and performance of the best (w.r.t the advantage value) synthesized control execution sequence for a suspension control system [14]. Our framework generates a

(a) Comparison Between $\triangle e^p$ and $\triangle e^{ap}$

(b) Comparison Between periodic y and aperiodic y under no FDI

Figure 6: Effect of Derived Aperiodic Control Execution Sequence $(1001)^\omega$ on Suspension Control System under FDI

minimum-length attack for this system which can make the system unsafe within 4 sampling iterations. The generated best possible attack-resilient control sequence is 1001 (the 1st sub-row in the 6-th column of the 4-th row of the Tab. 1). The blue plot with circle marker denotes $\triangle e^{ap}$ while following the control execution pattern 1001 and the red plot with square marker denotes $\triangle e^p$ while following periodic control execution. As we can clearly observe in Fig. 6a, under the 4-length attack $\triangle e^{ap}$ is significantly less than $\triangle e^p$. Due to this, the estimation error induced by the false data is significantly reduced by repeating the control execution skips at every 2-nd and 3-rd position of a 4 length execution/skip pattern. Fig. 6b showcases the performance of the system under the aperiodic control sequence 1001 (in blue and circled plot) and the periodic control execution (in 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 quickly compared to the aperiodic execution. However, since the aperiodic control sequences synthesized using our framework always follows the r_{min}, even under the

(a) TTC under minimum-length attack in periodic execution

(b) TTC under minimum-length attack under $10^6 1^6$

Figure 7: Effect of aperiodic control execution sequence on trajectory tracking control system under FDI attack

aperiodic execution the system output settles within 2.4seconds (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.

A Use case: Lightweight Security Design Utilizing Control Execution Sequence for Automotive CPS. Automotive CPSs are safety-critical but often resource-constrained. Thus we cannot afford to secure the closed loop communications in every iteration. The state-of-the-art technique to handle this resource-aware security design is to activate the cryptographic measures intermittently [5]. For this we can utilize the aperiodic control executions, synthesized using our framework, to make the system resilient enough against a minimum-length attack sequence while the cryptographic encryption is not active. As we have seen earlier, (refer Sec. 4.1) an attack vector generated using our framework commits maximum effort to make the system states unsafe without being detected in minimum possible time. But if we choose to run the closed loop control execution following the synthesized optimally attack-resilient control sequence, the attack effect (the estimation error difference Δe) can be kept in check for a longer duration. This enables the system to behave in a more resilient fashion under FDI attacks resulting to a less frequent activation of the cryptographic method. Let us visualize such a scenario. As we can see in Fig. 7a, under the periodic control execution sequence, outputs of the TTC i.e. distance from the desired trajectory (D in meters, the blue plots) and velocity of the vehicle (V in m/s, the red plots) goes beyond the safety boundaries at 13-th sampling iteration. But while following the best aperiodic execution pattern synthesized using our framework for TTC i.e. $10^6 1^6$, the system does not become unsafe at 13-th iteration. Rather as we can see in Fig. 7b, the generated minimum-length attack for $10^6 1^6$ is of 18 length, i.e. the attack makes the system unsafe at 18-th sampling iteration when operated with $10^6 1^6$. This simply suggests that the system under the synthesized aperiodic execution can promise more resilience against attack and can reduce the activation of the cryptographic method by $\sim 21\%$ (activation of crypto can be delayed from 14 sampling iterations in case of periodic executions to 17 in case of aperiodic executions). This motivates the fact such optimally attack-resilient control sequences can be useful in resource-aware CPS co-designs.

6 CONCLUSION

In this work, 1) we establish analytical conditions under which occasional control skips improve the system resilience w.r.t. FDI

attacks, and 2) provide an associated CAD framework for generating such skip sequences. Extending the constraint solving problem formulation (Eq. 10-16) in our methodology with a counter-example guided loop, it is possible to generate multiple attack vectors of minimum length and beyond the minimum length. For all such cases, applying the control execution sequence generation method provides a library of robust control schedules which can be deployed in a CPS. Creating a statistical foundation for choosing among such sequences given the probability distribution of attack vectors is considered as future work. Also, as the use case suggests, using attack-resilient control executions for relaxing real-time resource constraints in a methodical manner can be another future extension.

REFERENCES

[1] Sunandan Adhikary, Ipsita Koley, Saurav Kumar Ghosh, Sumana Ghosh, Soumyajit Dey, and Debdeep Mukhopadhyay. 2020. Skip to secure: Securing cyber-physical control loops with intentionally skipped executions. In *Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy*. 81–86.

[2] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. 2011. Comprehensive experimental analyses of automotive attack surfaces.. In *USENIX Security Symposium*, Vol. 4. San Francisco, 447–462.

[3] Sumana Ghosh, Souradeep Dutta, Soumyajit Dey, and Pallab Dasgupta. 2017. A structured methodology for pattern based adaptive scheduling in embedded control. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 1–22.

[4] Andy Greenberg. 2015. Hackers remotely kill a jeep on the highway-with me in it. *Wired* 7 (2015), 21.

[5] Ilija Jovanov and Miroslav Pajic. 2019. Relaxing integrity requirements for attack-resilient cyber-physical systems. *IEEE Trans. Automat. Control* 64, 12 (2019), 4843–4858.

[6] Ipsita Koley, Saurav Kumar Ghosh, Soumyajit Dey, Debdeep Mukhopadhyay, Amogh Kashyap KN, Sachin Kumar Singh, Lavanya Lokesh, Jithin Nalu Purakkal, and Nishant Sinha. 2020. Formal synthesis of monitoring and detection systems for secure cps implementations. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 314–317.

[7] Ralph Langner. 2013. To kill a centrifuge: A technical analysis of what stuxnet's creators tried to achieve. *The Langner Group* (2013).

[8] Vuk Lesi, Ilija Jovanov, and Miroslav Pajic. 2017. Security-aware scheduling of embedded control tasks. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 1–21.

[9] Rupak Majumdar, Indranil Saha, and Majid Zamani. 2011. Performance-aware scheduler synthesis for control systems. In *2011 Proceedings of the Ninth ACM International Conference on Embedded Software (EMSOFT)*. IEEE, 299–308.

[10] William C Messner, Dawn M Tilbury, and Rick Hill. 1999. Control tutorials for matlab and simulink. https://ctms.engin.umich.edu/CTMS/index.php?example=Suspension&section=SimulinkControl. Accessed: 2022-05-23.

[11] Yilin Mo and Bruno Sinopoli. 2010. False data injection attacks in control systems. In *Preprints of the 1st workshop on Secure Control Systems*. 1–6.

[12] Arslan Munir and Farinaz Koushanfar. 2018. Design and analysis of secure and dependable automotive CPS: A steer-by-wire case study. *IEEE TDSC* (2018).

[13] Jose Nazario. 2007. Blackenergy ddos bot analysis. *Arbor Networks* (2007).

[14] Debayan Roy, Licong Zhang, Wanli Chang, Dip Goswami, and Samarjit Chakraborty. 2016. Multi-objective co-optimization of FlexRay-based distributed control systems. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 1–12.

[15] Bastian Schürmann and Matthias Althoff. 2017. Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems. In *2017 American Control Conference (ACC)*. IEEE, 2522–2529.

[16] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. 2013. Non-invasive spoofing attacks for anti-lock braking systems. In *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 55–72.

[17] Jill Slay and Michael Miller. 2007. Lessons learned from the maroochy water breach. In *ICCIP*. Springer, 73–82.

[18] Damoon Soudbakhsh, Linh TX Phan, Oleg Sokolsky, Insup Lee, and Anuradha Annaswamy. 2013. Co-design of control and platform with dropped signals. In *Proceedings of the ACM/IEEE 4th international conference on cyber-physical systems*. 129–140.

[19] Andre Teixeira et al. 2015. Secure control systems: A quantitative risk management approach. *IEEE Control Systems Magazine* 35, 1 (2015), 24–45.

[20] Chee Tan Wei, Hazlina Selamat, and Ahmad Jais Alimin. 2009. Modeling and control of an engine fuel injection system. *Faculty of Electrical Engineering,*

University Teknologi Malaysia 1 (2009), 1–2.