Mobile Netw Appl (2020) 25:1944–1957
https://doi.org/10.1007/s11036-017-0943-5

Bio-inspired Active System Identification: a Cyber-Physical Intelligence Attack in Networked Control Systems

Alan Oliveira de Sá[1,2] · **Luiz F. R. da C. Carmo**[2,3] · **Raphael C. S. Machado**[3,4]

Published online: 5 October 2017

Abstract From the point of view of the control theory, the literature indicates that stealthy and accurate cyber-physical attacks on Networked Control System (NCS) must be planned based on an accurate knowledge about the model of the attacked system. However, most literature about these attacks does not indicate how such knowledge is obtained by the attacker. So, to fill this hiatus, an Active System Identification attack is proposed in this paper, where the attacker injects data on the NCS to learn about its model. The attack is implemented based on two bio-inspired metaheuristics: Backtracking Search Optimization Algorithm (BSA) and Particle Swarm Optimization (PSO). To improve the accuracy of the estimated models, a statistical refinement is proposed for the outcomes of the two optimization algorithms. Additionally, a set of data injection attacks are shown in order to demonstrate the capability of the proposed attack in supporting the design of other sophisticated attacks. The results indicate a better performance of the BSA-based attacks, especially when the captured signals contain white Gaussian noise. The goal of this paper is to demonstrate the degree of accuracy that this System Identification attack may achieve, highlighting the potential impacts and encouraging the research of possible countermeasures.

Keywords Security · Cyber-physical systems · Networked control systems · System identification · Backtracking search algorithm · Particle swarm optimization

This research was partially supported by the Brazilian research agencies CNPq and FAPERJ.

✉ Alan Oliveira de Sá
alan.oliveira.sa@gmail.com

Luiz F. R. da C. Carmo
lfrust@inmetro.gov.br

Raphael C. S. Machado
rcmachado@inmetro.gov.br

[1] Admiral Wandenkolk Instruction Center, Brazilian Navy, Rio de Janeiro, Brazil

[2] Institute of Mathematics/NCE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

[3] National Institute of Metrology, Quality and Technology, Rio de Janeiro, Brazil

[4] Rio de Janeiro Federal Center for Technological Education, Rio de Janeiro, Brazil

1 Introduction

System identification, i.e. the action of building mathematical models of dynamic systems, is often used to obtain the model of physical processes aiming to support the design of their respective control systems. However, it can also be considered a key step for the execution of accurate and stealth – or covert, as mentioned in [7, 20, 21, 24] – attacks against Networked Control Systems (NCS). Indeed, to reduce the probability to be detected by algorithms that monitor the dynamics of the controlled plant, the attacker must have an accurate model of the targeted system, such as demonstrated in [21, 24].

A possible strategy to obtain information about the model of the targeted system is through passive System Identification attacks, as reported in [7]. In that technique, the attacker eavesdrops the communications between the controller, actuators and sensors of the NCS until enough information is collected to determine the parameters of the plant and its control system. Such passive approach can

make the system identification more time consuming, until meaningful information transits at the eavesdropped communication links. The situation is even worse if the system is in steady state because no meaningful information may transit through the NCS's communication links for a long time. The information content of the signals measured under steady operating conditions is often insufficient for identification purposes [26]. This attacker's constraint may be overcome by the Active System Identification Attack herein proposed, which, as far as we know, is not reported in the literature.[1]

Our approach was inspired by the classic active cryptanalytic attacks (chosen plaintext and chosen cyphertext), where the attacker inserts messages in the crypto-engine to deduce the secret key. Note that this is the opposite of the passive attacks (cyphertext only and known plaintext), where the attacker simply listens the communication channels and passively collects information to recover the secret key [23].

In the attack presented in this work, a specially tailored signal is inserted by the attacker in an NCS communication channel. After that, by observing the behavior of the system in closed-loop, the attacker determines the parameters of its open-loop transfer function. To do so, the attacker only needs to intercept one communication channel of the NCS, where the attacker both inserts the attack signal and listens the consequent system response.

If an attack signal $a(k)$ and the consequent response $y_a(k)$ of an NCS are known, the open-loop transfer function can be assessed by applying $a(k)$ in an estimated model, which is adjusted until its estimated output $\hat{y}_a(k)$ matches $y_a(k)$. The Backtracking Search Optimization algorithm (BSA) [4] and the Particle Swarm Optimization (PSO) [12] are herein used to iteratively adjust the parameters of the estimated model, by minimizing a specific fitness function until the estimated model converges to the actual model of the NCS. The BSA and the PSO are chosen to perform this task due to their capability to converge to good solutions, such as demonstrated in [10, 11, 16, 27] specifically for control system problems. Given the stochastic nature of the used algorithms (BSA and PSO), the results need to be statistically analyzed in order to perform a refinement of the estimated model. In this work, the statistical refinement used in [6] is improved, leading to a more accurate estimated model than the results obtained in [6].

The knowledge of the NCS's open-loop transfer function obtained by the Active System Identification attack is useful for the design of other sophisticated attacks [7, 21]. To demonstrate this usefulness from the attacker point of view, this paper includes the simulation of a set of data injection attacks designed based on the data gathered by the Active System Identification attack. In these simulations, not presented in [6], the attacker accurately induces an overshoot on the attacked plant, which may cause stress and possible damages [8, 25], reducing its mean time between failure (MTBF).

It is worth mentioning that the Active System Identification attack herein proposed is different from the active attacks performed to identify vulnerabilities of protocols and applications within the layers of the OSI model, such as the active scanning process used to identify network services [2]. The attack herein proposed aims to identify the physical model of a plant that, in an NCS, lies above the application layer of the OSI model.

Note that the applications of NCSs can range from cooperative control of vehicles using mobile networks [15, 17] to large Pressurized Heavy Water Reactors (PHWR) [5] or water canal systems [1, 21] controlled by wired NCSs. This include a vast number of potential – sometimes critical – targets for the attack herein proposed. In this sense, the goal of this paper is to demonstrate the degree of accuracy that the present attack may achieve, highlighting its potential impacts and encouraging the research of countermeasures capable to prevent or detect its execution.

In summary, the main contributions of this paper, with regard to the preliminary version of this work [6], are:

- The review on the taxonomy presented in [7], in order to encompass the Active System Identification attack in the context of the Cyber-Physical Intelligence (CPI) attacks. Also, it sets the role that the proposed attack – as a CPI attack – plays in building model-dependent attacks.
- The proposal of a new statistical refinement method for the outcomes provided by the bio-inspired metaheuristics. The results demonstrate that this refinement improves the quality of the information produced by the identification attack.
- The novel joint operation of the Active System Identification Attack and a Controlled Data Injection Attack, which allows the evaluation on how a model-dependent attack can benefit from the intelligence obtained by the Active System Identification Attack. The results indicate that the referred model-dependent attack can achieve high accuracy when supported by the Active System Identification Attack, specially when the latter is statistically refined by the method introduced in this paper.

The remainder of this paper is organized as follows. In Section 2, we review the literature of NCS attacks, with focus on the intelligence gathered to support their design.

[1] A preliminary version of this work was presented in the 10th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT 2017) and published in the proceedings of the event [6]. The present paper proposes a refinement for the system identification method described in [6] and simulates a data injection attack using the data obtained after this refinement.

In Section 3, we discuss and review the taxonomy presented in [7] in order to encompass the attack herein proposed. In Sections 4 and 5, we provide brief descriptions of the BSA and PSO, respectively. In Section 6, the Active System Identification attack, herein proposed, is described. Section 7 presents the results achieved by the proposed attack, comparing both metaheuristics in simulations where the NCS is constituted by a DC motor and a proportional-integral (PI) controller. Also, Section 7 quantitatively demonstrates the accuracy that a data injection attack may achieve, when supported by the proposed Active System Identification attack. Section 8 contains our final considerations.

2 Related works

The possibility of large impact cyber-physical attacks became unprecedentedly concrete after the launch of the Stuxnet worm [13] and has been motivating researches concerning the security of NCSs. In this section, a review of the literature related to this subject is presented.

In [14] the authors propose two queuing models that are used to evaluate the impact of delay jitter and packet loss in an NCS under attack. The attack is not designed taking into account the models of the controller and the physical plant. Such models are unknown by the attacker. Thus, to affect the plant's behavior, the attacker arbitrarily floods the network with traffic, causing jitter and packet loss. In this method of attack, the excess of packets in the network can reduce the stealthiness of the attack, allowing the adoption of countermeasures, such as packet filtering [14] or blocking the malicious traffic on its origin [22]. Moreover, the arbitrary intervention in a system which the models are unknown may lead the plant to an extreme physical behavior, which is not desired if a stealth attack is intended [7].

In [9], a testbed for Supervisory Control and Data Acquisition (SCADA) using TrueTime (a MATLAB/ Simulink based tool) is presented. The authors demonstrate an attack where a malicious agent transmits false signals to the controller and actuator of an NCS. The false signals are randomly generated, aiming to make a DC motor lose its stability. This kind of attack does not require a previous knowledge about the plant and controller of the NCS. The drawback is that the desired physical effect and the stealthiness of the attack cannot be ensured due to unpredictable consequences from the application of random false signals to a system which the model is not known.

A general framework for the analysis of a wide variety of attacks over NCSs is provided in [24]. The authors classify and establish the requirements for the attacks in terms of model knowledge, disclosure and disruption resources. In their work, it is stated that covert attacks require high level of knowledge about the model of the targeted system. Examples of covert attacks that agree with this statement are provided in [20, 21]. In these works, the attacks are performed by a man-in-the-middle (MitM), where the attacker needs to know the model of the plant under attack and also inject false data in both forward and feedback streams. The stealthiness of the attacks described in [20, 21] is analyzed from the perspective of the signals arriving to the controller and depends on the difference between the actual model of the plant and the model known by the attacker. In [1], another stealth attack is demonstrated. The attacker, aware of the system's model, injects an attack signal in the NCS to steal water from the Gignac canal system located in Southern France.

In [1, 20, 21, 24], where a previous knowledge about the models of the NCS under attack is required, it is not described how this knowledge is obtained by the attacker. It is just stated that a model is previously known to support the design of the attack. More recently, in [7], the authors propose a System Identification attack to fill this hiatus. They demonstrate how the data required for the design of Denial-of-Service (DoS) or Service Degradation (SD) attacks may be obtained through a passive System Identification attack. The attack proposed in [7] does not need to inject signals on the NCS to estimate its models. However, it depends on the occurrence of events, that are not controlled by the attacker, to produce signals that carry meaningful information for the system identification algorithm. The Active System Identification attack, herein proposed, constitutes an alternative to the passive System Identification attacks in situations where the attacker may not wait so long for the occurrence of such meaningful signals. A synthesis of the characteristics of the attacks referred in this section is presented in Table 1.

3 Taxonomy

In [7], the authors propose a taxonomy that encompasses three main classes of attack – Denial-of-Service (DoS), Service Degradation (SD), and Cyber-Physical Intelligence (CPI) – in which the service to be attacked/ protected is the work performed by the physical process controlled by an NCS. According to that taxonomy, the DoS attacks are intended to interrupt the execution of the work performed by the controlled plant, or even destroy the plant in a short term. On the other hand, the SD attacks aims to reduce the efficiency of the physical process, or even reduce the mean time between failure (MTBF) of the plant in mid/long term. Yet, according to that taxonomy, the CPI attacks are intended to gather information of the NCS basically through two kinds of attack – eavesdropping, and System Identification attacks –, in order to provide the information necessary for planning and designing DoS and SD controlled attacks. The referred taxonomy establishes the requirements for each attack of

Table 1 Synthesis of the related attacks

Attack	Method	System knowledge	How the knowledge is obtained
Stuxnet *worm* [13]	Modifications in the PLC code	Yes	Experiments in a real system
Long, et al. [14]	Inducing *jitter* and packet loss	None	N/A
Farooqui, et al. [9]	Data injection	None	N/A
Smith [20, 21]	Data injection	Yes	Not described
Teixeira [24]	Packet loss	None	N/A
	Data injection	Yes	Not described
Amin [1]	Data injection	Yes	Not described
SD-Controlled [7]	Data injection	Yes	Passive system identification

these three main classes and, above all, explains how model-dependent attacks, such as the DoS and SD controlled attacks can benefit from the information provided by CPI attacks.

The attacks belonging to the first two classes, i.e. DoS and SD, are premised active, once they act through the induction of jitter, data loss or data injection on the NCS. On the other hand, according to that taxonomy, the attacks belonging to the CPI class of attacks do not impact or interfere on the NCS, once they only need to listen the control signals that flow through the NCS. The eavesdropping attack simply capture the control signals that flow through the network. The System Identification attack, according to [7], collects the data that flows through the input and output of the NCS devices, i.e. controllers and plants, and uses the collected information to passively estimate the model of such devices.

However, the results achieved by the present work lead us to review the taxonomy proposed in [7], specifically with regard to the System Identification attacks. Different from the System Identification attack defined in [7], the attack proposed in this paper requires the injection of an attack signal in the NCS, in order to estimate its model through the analysis of its consequent response. Thus, it is necessary to expand the taxonomy related to System Identification attacks, that are now divided within two kinds, as shown in Fig. 1:

– The Passive System Identification attacks: this kind of attack estimates the model of an NCS based on the analysis of the signals collected from the input and output of the system's devices. This kind of attack analyzes signals that typically flow through the NCS, as a result of its normal operation. In this case, both input and

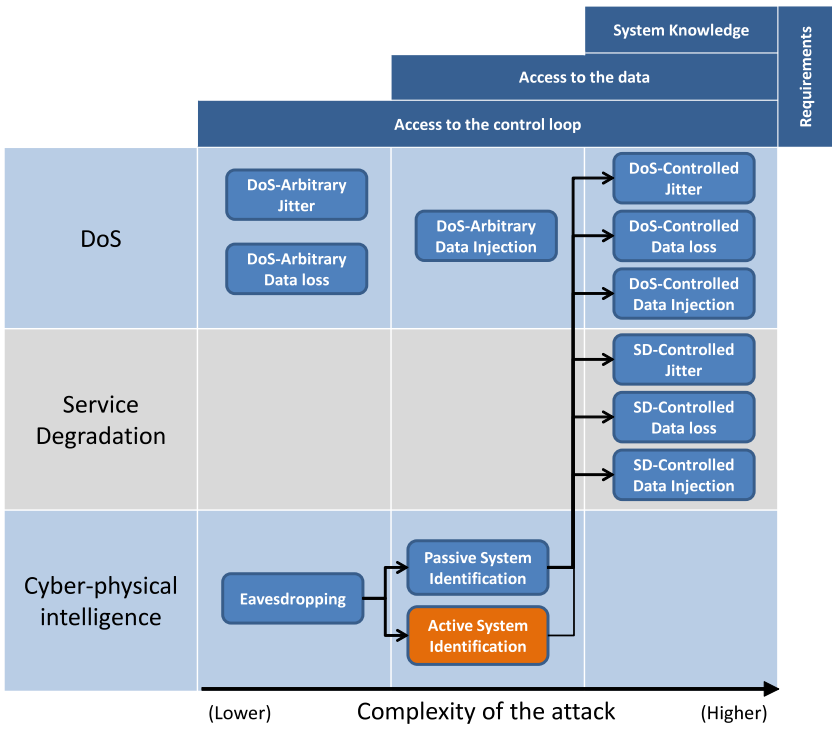


Fig. 1 Classification and requirements of the cyber-physical attacks that act in the control loop of an NCS

output signals must carry meaningful information – i.e. information enough to estimate the transfer function of the attacked system/device –, and it is not necessary to inject signals into the attacked system.
- The Active System Identification attack: in this kind of attack, aim of this work, the attacker injects a signal into the system and estimates its model based on the system's response in face of the attack signal. From the attacker point of view, this attack is useful, for example, when the system is in steady state and the attacker cannot wait for a signal carrying meaningful information for the identification process.

It is noteworthy that an Active System Identification attack is less stealthy than a Passive System Identification attack, given that the former needs to interfere in the system and the latter just needs to listen its signals. In this sense, when performing an Active System Identification attack, the attacker must choose signals that, when injected on the NCS, are more difficult to be perceived by a defense system. From the defender perspective, it is important to be aware of this kind of attack and also learn about the stealthiness of Active System Identification attacks, in order to develop techniques to identify and avoid them.

4 Backtracking search algorithm

In this section, the basic concepts of the BSA are described in order to provide a clear comprehension regarding to the parameters of the algorithm that are adjusted for the attack. The BSA is a bio-inspired metaheuristic that searches for solutions of optimization problems using the information obtained by past generations – or iterations. According to [4], its search process is metaphorically analogous to the behavior of a social group of animals that, at random intervals, returns to hunting areas previously visited for food foraging. The general evolutionary structure of the BSA is shown in Algorithm 1.

Algorithm 1 BSA

```
begin
    Initialization;
    repeat
        Selection-I;
        Generate new population
            Mutation;
            Crossover;
        end
        Selection-II;
    until Stopping Condition;
end
```

At the Initialization stage, the algorithm generates and evaluates the initial population $\mathcal{P}_0$ and sets the historical population $\mathcal{P}_{hist}$. The latter constitutes the BSA's memory that, in Selection-I stage, is updated with historical coordinates visited by the individuals.

During the first selection stage (Selection-I), the algorithm randomly determines, based on an uniform distribution U, whether the current population $\mathcal{P}$ should be kept as the new historical population and, therefore, replace $\mathcal{P}_{hist}$ (i.e. if $a < b|a, b \sim U(0, 1)$, then $P_{hist} = P$). Subsequently, at every iteration, it shuffles the individuals of $\mathcal{P}_{hist}$ – having $\mathcal{P}_{hist}$ been replaced or not.

The mutation operator creates $\mathcal{P}_{mod}$, which is the preliminary version of the new population $\mathcal{P}_{new}$). It does so according to Eq. 1:

$$\mathcal{P}_{mod} = \mathcal{P} + \eta \cdot \Gamma(\mathcal{P}_{hist} - \mathcal{P}), \tag{1}$$

wherein η is empirically adjusted through simulations and $\Gamma \sim N(0, 1)$, with N being a normal standard distribution. Therefore, $\mathcal{P}_{mod}$ is the result of the movement of $\mathcal{P}$'s individuals in the directions established by vector $(\mathcal{P}_{hist} - \mathcal{P})$ and η controls the displacements' amplitude.

In order to create the final version of $\mathcal{P}_{new}$, the crossover operator randomly combines, also following a uniform distribution, individuals from $\mathcal{P}_{mod}$ and others from $\mathcal{P}$.

At the second selection stage (Selection-II), the algorithm firstly evaluates the individuals of $\mathcal{P}_{new}$ using a fitness function f. After that, individuals of $\mathcal{P}$ (i.e. individuals before applying the mutation and crossover operators) are replaced by individuals of $\mathcal{P}_{new}$ (i.e. individuals obtained after mutation and crossover) with better fitness. Therefore, $\mathcal{P}$ includes only new individuals that evolved. While the stopping condition has not yet been reached, the algorithm iterates. Otherwise, it returns the best solution found.

Note that the algorithm has two parameters that are empirically adjusted: the size $|\mathcal{P}|$ of its population $\mathcal{P}$; and η, that establishes the amplitude of the movements of the individuals of $\mathcal{P}$. The parameter η must be adjusted to assign good exploration and exploitation capabilities to the algorithm. With these parameters set, the BSA is used to search for the global minimum of the fitness function described in Section 6.

5 Particle swarm optimization

PSO has roots in the collective behavior of social models such as bird flocking and fish schooling. A particle, i.e. the basic element of the algorithm, represents a possible solution of a problem. Therefore, the swarm represents a set of possible solutions. At each iterative cycle, the position of

each particle is updated according to Eq. 2, where x_j and v_j are the position and velocity of particle j, respectively.

$$x_j(t+1) = x_j(t) + v_j(t+1) \tag{2}$$

The computation of v_j considers three terms: the particle's inertia; the particle's cognition, which is based on the best solution found by the particle so far; and social term, which is based on global best solution found by the swarm. The velocity of particle j, at each dimension d, is defined in Eq. 3:

$$\begin{aligned} v_{jd}(t+1) = {} & \omega v_{jd}(t) + \varphi_1 r_{1d}(t)(m_{jd} - x_{jd}(t)) \\ & + \varphi_2 r_{2d}(t)(m_{gd} - x_{jd}(t)), \end{aligned} \tag{3}$$

wherein ω is a parameter that weighs the inertia of the particle, φ_1 and φ_2 are parameters that weigh the cognitive and social terms, respectively, r_1 and r_2 are random numbers in [0,1], m_j is the best position visited by particle j so far, and m_g is the best position discovered by the swarm considering the experience of all the particles. To obtain m_j and m_g the algorithm evaluates, at each iteration, the position x_j of each particle j using a fitness function $f(x)$.

In order to better explore multi-dimensional search spaces, a velocity limit is imposed for each dimension d, as in Eq. 4:

$$0 \leq v_{jd} \leq \delta(max_d - min_d), \tag{4}$$

wherein max_d and min_d are the maximum and minimum limits of the search space at each dimension d and $\delta \in [0, 1]$. The overall computation that the PSO performs to minimize a fitness function $f(x)$ is given in Algorithm 2, where x is the particle position and $\mathcal{S}$ is the swarm size.

Algorithm 2 PSO algorithm

```
begin
    for each particle j, 1 ≤ j ≤ S do
        Set randomly position x_j and velocity v_j;
        m_j ← x_j;
    end
    m_g ← smallest m_j, 1 ≤ j ≤ S;
    repeat
        for each particle j, 1 ≤ j ≤ S do
            Update velocity v_j, as in Eqs. 3 and 4;
            Update position x_j, as in Eq. 2;
            fitness ← f(x_j);
            m_k ← x_j, whenever fitness < f(m_j);
            m_g ← x_j, whenever fitness < f(m_g);
        end
    until Stopping condition;
    return m_g;
end
```

6 The active system identification attack

The Active System Identification attack, herein proposed, is intended to assess the coefficients of a transfer function $G(z) = C(z)P(z)$ of an NCS, wherein $C(z)$ is the controller's control function and $P(z)$ is the plant's transfer function, as shown in Fig. 2. The transfer functions are all linear time-invariant (LTI). This attack is performed by a MitM that may be located either in the forward or in the feedback link. For the sake of clarity of the analysis presentation, but without loss of generality, we focus on the case where the MitM is in the feedback link, i.e. between the plant's sensors and the controller's input. To estimate the model of the attacked NCS, the attacker injects an attack signal $a(k)$ and measures the response of the system to such signal.

The complete response of the generic NCS shown in Fig. 2, considering only the inputs $R(z) = \mathcal{Z}[r(k)]$ and $A(z) = \mathcal{Z}[a(k)]$, is expressed in the z domain by Eq. 5:

$$Y(z) = \frac{G(z)}{1+G(z)}R(z) - \frac{G(z)}{1+G(z)}A(z), \tag{5}$$

wherein $Y(z) = \mathcal{Z}[y(k)]$. $\mathcal{Z}$ represents the Z-transform operation. As a premise, in a normal condition, it is considered that $a(k) = 0$ and the system is designed to make $y(k) \to q$, in such way that $y(k) \approx q \forall k > k_s$, i.e. the output $y(k)$ of the NCS converges and stabilizes at a constant value q after a certain amount of samples k_s. Indeed, it is usually one of the main aims of a control system. Now, considering $a(k) \neq 0$, the output $y(k), \forall k > k_s$, may be defined approximately as Eq. 6:

$$y(k) = q - \mathcal{Z}^{-1}\left[\frac{G(z)}{1+G(z)}A(z)\right], \forall k > k_s. \tag{6}$$

Thus, after k_s, the portion of $y(k)$ caused by $r(k)$ can be eliminated by just subtracting q from Eq. 6, which leads to Eq. 7:

$$y_a(k) = y(k) - q = -\mathcal{Z}^{-1}\left[\frac{G(z)}{1+G(z)}A(z)\right], \forall k > k_s. \tag{7}$$

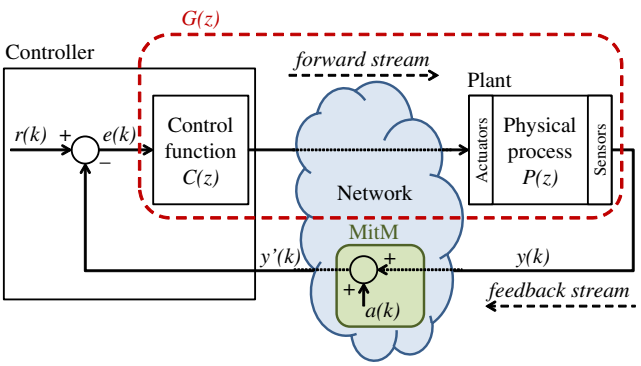


Fig. 2 Active System Identification attack with a MitM in the feedback link

wherein $y_a(k)$ represents the portion of $y(k)$ caused by the attack signal $a(k)$. The value of q can be assessed by the attacker through an eavesdropping attack in the feedback stream, by just capturing $y(k)$ after the stabilization of the NCS. The subtraction of q after k_s makes the system identification attack independent of $r(k)$ $\forall k > k_s$. The Active System Identification attack now just relies on the attack signal $a(k)$, which can be chosen, and the response of the system to the attack $y_a(k)$ can be obtained in accordance with Eq. 7. The signal $y_a(k)$ starts with the injection of $a(k)$ and has the size of a monitoring period T.

If the attack input $a(k)$ and its consequent output $y_a(k)$ are known, the model of $G(z)$ can be assessed by applying the known $a(k)$ in an estimated system, defined by Eq. 8:

$$\hat{y}_a(k) = -\mathcal{Z}^{-1}\left[\frac{G_e(z)}{1+G_e(z)}\right] * a(k), \tag{8}$$

wherein $G_e(z)$ is the estimation of $G(z)$ and $\hat{y}_a(k)$ is the output of the estimated system in face of $G_e(z)$. By comparing $\hat{y}_a(k)$ with $y_a(k)$, the attacker is capable to evaluate whether $G_e(z)$ is equal/approximately $G(z)$. Note that $G_e(z)$ is a generic transfer function represented by Eq. 9:

$$G_e(z) = \frac{\alpha_n z^n + \alpha_{n-1} z^{n-1} + \ldots + \alpha_1 z^1 + \alpha_0}{z^m + \beta_{m-1} z^{m-1} + \ldots + \beta_1 z^1 + \beta_0}, \tag{9}$$

wherein n and m are the order of the numerator and the denominator, respectively, while $[\alpha_n, \alpha_{n-1}, \ldots \alpha_1, \alpha_0]$ and $[\beta_{m-1}, \beta_{m-2}, \ldots \beta_1, \beta_0]$ are the coefficients of the numerator and the denominator, respectively, that are intended to be found by this Active System Identification attack. Therefore, to find $G(z)$, the coefficients of $G_e(z)$ are adjusted until the estimated output $\hat{y}_a(k)$ converges to the known $y_a(k)$.

In this sense, the BSA and the PSO are used to iteratively adjust the estimated model, by minimizing a specific fitness function presented in this section until the estimated model $G_e(z)$ converges to the actual $G(z)$ of the real NCS. To compute the fitness of the individuals of the optimization algorithm (i.e. the BSA or PSO), the same attack signal $a(k)$ that caused $y_a(k)$ is applied on the estimated system defined by Eqs. 8 and 9, where the coefficients of $G_e(z)$ are the coordinates $x_j = [\alpha_{n,j}, \alpha_{n-1,j}, \ldots \alpha_{1,j}, \alpha_{0,j}, \beta_{m-1,j}, \beta_{m-2,j}, \ldots \beta_{1,j}, \beta_{0,j}]$ of an individual j of the BSA/PSO. The output $\hat{y}_{aj}(k)$ is the response of the estimated model (8, 9) in face of $a(k)$, when the coefficients of $G_e(z)$ are x_j. Then, the fitness f_j of each individual j is obtained comparing $\hat{y}_{aj}(k)$ with $y_a(k)$, according to Eq. 10:

$$f_j = \frac{\sum_{k=0}^{N} (y_a(k) - \hat{y}_{aj}(k))^2}{N}, \tag{10}$$

wherein N is the number of samples that exist during the monitoring period T of $y_a(k)$. Note that, if no other inputs – perturbation or noise – occur in the NCS during T, then $\min f_j = 0$ when $[\alpha_{n,j}, \alpha_{n-1,j}, \ldots \alpha_{1,j}, \alpha_{0,j}, \beta_{m-1,j}, \beta_{m-2,j}, \ldots \beta_{1,j}, \beta_{0,j}] = [\alpha_n, \alpha_{n-1}, \ldots \alpha_1, \alpha_0, \beta_{m-1}, \beta_{m-2}, \ldots \beta_1, \beta_0]$, i.e. when the estimated $G_e(z)$ converges to $G(z)$.

An analogy may be established between this Active System Identification attack and the Chosen Plaintext cryptanalytic attack [23], wherein $a(k)$ corresponds to the chosen plaintext, $y_a(k)$ represents the ciphertext, the Eqs. 8 and 9 together correspond to the encryption algorithm, and the actual coefficients $[\alpha_n, \alpha_{n-1}, \ldots \alpha_1, \alpha_0]$ and $[\beta_{m-1}, \beta_{m-2}, \ldots \beta_1, \beta_0]$ of $G_e(z)$ correspond to the secret key.

It is worth mentioning that this attack requires the previous knowledge about the order of the numerator and denominator of Eq. 9 (n and m, respectively). Using the analogy with the Chosen Plaintext cryptanalytic attack, it is equivalent to require the knowledge about the size of the secret key of the encryption algorithm. In this Active System Identification attack, the information of n and m is necessary to define the number of dimensions of the search space of the BSA – or the number of unknown coefficients of $G(z)$ – which must be set to $n + m - 1$. Although this is a constraint of the attack, this information may be inferred if the attacker, at least, knows what the attacked plant is and what type of controller is being used.

7 Results

In Section 7.1, the results obtained by both BSA-based and PSO-based Active System Identification attacks are analyzed and statistically refined in order to provide a demonstration of the degree of accuracy that the attacker may obtain with the proposed attack. Additionally, Section 7.2 presents a set of data injection attacks designed based on the models estimated by the Active System Identification process. The purpose of the simulations of these data injection attacks is to demonstrate how an Active System Identification attack may contribute for the accuracy of other sophisticated attacks.

7.1 Active system identification attack

The attacked system, shown in Fig. 3, consists of a DC motor whose rotational speed is controlled by a Proportional-Integral (PI) controller. This example is chosen due to the use of DC motors in a vast number of real world control systems. Moreover, DC motors has been widely used in previous works about NCS [3, 7, 14, 18, 19]. It is noteworthy that the model herein chosen as an example does not exhaust the potential targets for this attack. NCSs composed by another kinds of LTI devices may also be a target.

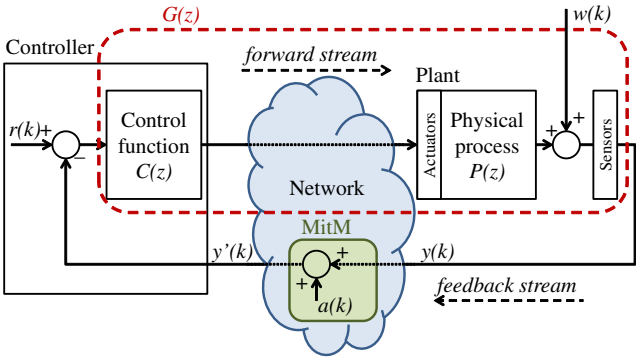


Fig. 3 Attack on a noisy NCS

However, it must be taken into account that the computational cost of the attack, when launched over different LTI systems, may vary with the number of their unknown coefficients – i.e. the number of dimensions of the search space explored by the optimization algorithms (BSA or PSO, in this paper).

The PI control function $C(z)$ and the DC motor transfer function $P(z)$, obtained from [14], are represented by Eqs. 11 and 12, respectively:

$$C(z) = \frac{0.1701z - 0.1673}{z - 1}, \tag{11}$$

$$P(z) = \frac{0.3379z + 0.2793}{z^2 - 1.5462z + 0.5646}. \tag{12}$$

Thereby, the transfer function to be identified $G(z)$ – which is also the open-loop transfer function of the NCS – is defined by Eq. 13:

$$G(z) = C(z)P(z) = \frac{g_1 z^2 + g_2 z + g_3}{z^3 + g_4 z^2 + g_5 z + g_6}, \tag{13}$$

wherein $g_1 = 0.0575$, $g_2 = -0.0090$, $g_3 = -0.0467$, $g_4 = -2.5462$, $g_5 = 2.1108$ and $g_6 = -0.5646$. The sample rate of the system is 50 samples/s and the set point $r(k)$ is an unitary step function. Network delay and packet loss are not taken into account in the simulations of this paper.

The structure of the Eqs. 11 and 12, and so the structure of Eq. 13, are previously known by the attacker once that, as a premise, it is known that the target is an NCS that controls a DC motor using a PI controller. Thus, in these simulations, the goal of the Active System Identification attack is to discover g_1, g_2, g_3, g_4, g_5 and g_6.

The chosen attack signal $a(k)$ is a discrete-time unit impulse (14):

$$a(k) = \begin{cases} 1 \text{ if } k = k_a; \\ 0 \text{ otherwise}, \end{cases} \tag{14}$$

wherein k_a is the single sample in which the attacker interfere in the system by adding 1 to the feedback stream. Note that the discrete-time unit impulse is chosen to excite the NCS due to its short active time – i.e. one sample –, which increases the stealthiness of the attack in the time domain. Moreover, the Fourier transform of an impulse function has an uniform – flat – density in the frequency domain , which is easily masked by the frequency distribution of a white Gaussian noise. This fact also increases the stealthiness of the attack signal in the frequency domain.

The effectiveness of the Active System Identification attack is evaluated with and without noise. To simulate the noise, $w(k) \sim N(\mu, \sigma)$ is inserted in the NCS as indicated in Fig. 3. Note that $w(k)$ is a white Gaussian noise wherein N is a normal distribution, μ is its mean and σ is its standard deviation. In all simulations, the mean is $\mu = 0$ rad/s. The standard deviation is adjusted in such manner that 95% of the amplitudes of $w(k)$ are within $\pm I$ ($I = 2\sigma$). The simulations consider four different noise intensities I: 0 (no noise), 0.0025 rad/s, 0.005 rad/s and 0.01 rad/s. For each noise intensity I, 100 different simulations are executed using each of the mentioned metaheuristics. In each simulation, the feedback stream is captured by the attacker during a period $T = 2s$ (100 samples), starting at sample $k_a + 1$.

The attack model was implemented in MATLAB, where the simulations were carried out. The SIMULINK tool was used to compute $y_a(k)$ and $\hat{y}_{aj}(k)$ – the latter, for each individual j of the optimization algorithms. The parameters of the BSA and PSO described in Sections 4 and 5, respectively, were empirically adjusted through a set of simulations without noise ($I = 0$). These parameters are then used for all noise conditions. In the BSA-based attacks, the parameter η is set to 1. In the PSO-based attacks, the following parameters configuration is used: $\omega = 0.4$, $\varphi_1 = \varphi_2 = 1.5$ and $\delta = 0.1$. In both algorithms, the population is set to 100 individuals and the limits of each dimension of the search space are $[-10, 10]$. In each simulation, the BSA and the PSO are executed for 4500 iterations.

Let S_u be the solution of an attack simulation u, and $g_{i,u}$ the value estimated for the i^{th} coefficient of $G(z)$ in the u^{th} attack simulation. Each attack simulation provides a solution $S_u = [g_{1,u}, g_{2,u}, g_{3,u}, g_{4,u}, g_{5,u}, g_{6,u}]$ containing estimated values for the six coefficients of $G(z)$. In [6], for a given coefficient g_i of $G(z)$, if an estimated value $g_{i,u}$ is beyond two standard deviation from the mean, then $g_{i,u}$ is considered an outlier and eliminated from the set of values found for g_i. After that, the estimated value of each g_i is assumed to be the mean of the remaining $g_{i,u}$. However, in the present work, to improve the accuracy of the estimated model, this statistical refinement is modified. In this paper, if an estimated value $g_{i,u}$ is beyond two standard deviation from the mean, the whole solution S_u (to which $g_{i,u}$ belongs) is considered as an outlier and eliminated from the set of solutions. Doing so, the estimated value of each g_i is assumed to be mean of all $g_{i,u}$ contained in the set of

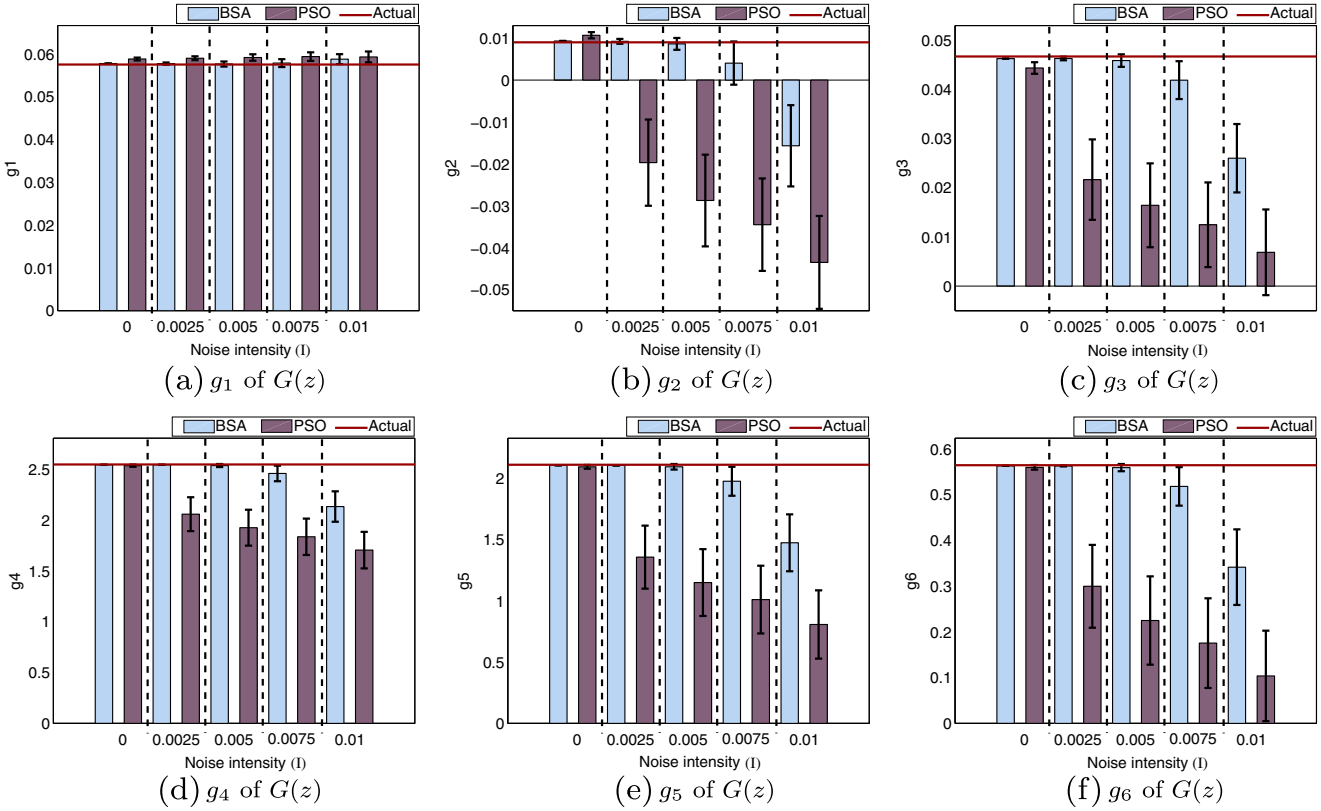
Table 2 Mean estimated coefficients of $G(z)$ after the statistical refinement

		Mean of the coefficients statistically refined in [6]						Mean of the coefficients statistically refined in the present work					
	Noise (I)	$g_1(\times 10^{-2})$	$g_2(\times 10^{-3})$	$g_3(\times 10^{-2})$	g_4	g_5	$g_6(\times 10^{-3})$	$g_1(\times 10^{-2})$	$g_2(\times 10^{-3})$	$g_3(\times 10^{-2})$	g_4	g_5	$g_6(\times 10^{-1})$
BSA	0	5.7756	−9.3337	−4.6261	−2.5431	2.1063	−5.6319	5.7750	−9.3128	−4.6268	−2.5431	2.1063	−5.6319
	0.0025	5.7736	−9.2001	−4.6301	−2.5428	2.1058	−5.6305	5.7714	−9.2299	−4.6294	−2.5428	2.1059	−5.6306
	0.005	5.7826	−9.0411	−4.5528	−2.5345	2.0937	−5.5924	5.7628	−8.6145	−4.5870	−2.5350	2.0944	−5.5931
	0.0075	5.8215	−0.7908	−3.4930	−2.4023	1.8911	−4.7857	5.7843	−4.0346	−4.1886	−2.4578	1.9761	−5.1824
	0.01	5.8561	20.7982	−2.5371	−2.0906	1.3852	−3.1095	5.8763	15.6817	−2.6009	−2.1322	1.4738	−3.4164
PSO	0	5.8799	−10.6784	−4.4361	−2.5341	2.0940	−5.5989	5.8799	−10.6784	−4.4361	−2.5341	2.0940	−5.5989
	0.0025	5.8987	19.7038	−2.1653	−2.0568	1.3567	−2.9982	5.8987	19.7038	−2.1653	−2.0568	1.3567	−2.9982
	0.005	5.9148	28.7309	−1.6431	−1.9242	1.1493	−2.2507	5.9148	28.7309	−1.6431	−1.9242	1.1493	−2.2507
	0.0075	5.9357	34.5026	−1.2472	−1.8347	1.0102	−1.7552	5.9357	34.5026	−1.2472	−1.8347	1.0102	−1.7552
	0.01	5.9288	43.4950	−0.6878	−1.7036	0.8073	−1.0370	5.9288	43.4950	−0.6878	−1.7036	0.8073	−1.0370

remaining S_u. Table 2 presents a summary that compares the results achieved in this work with the results obtained in [6], in both BSA-based and PSO-based attacks. The most accurate results are highlighted. Note that in all cases the most accurate results were achieved by the BSA-based attacks. According to Table 2, the statistical refinement used in the present work in general improves the accuracy of the results obtained by the BSA-based attacks. This improvement is more evident in Section 7.2, where the performance of other attacks designed with the data presented in Table 2 is analyzed. Note that the results shown in Table 2 for the PSO-based attacks are the same as the results of [6]. This occurs because in PSO-based attacks all outlier coefficients belong to solutions wherein all other coefficients are also outliers – i.e. beyond two standard deviations from their means. Thus, in the PSO-based attacks, the whole solution S_u which contains an outlier is eliminated from the set of solutions even when the statistical refinement of [6] is applied.

The mean estimated values of g_1, g_2, g_3, g_4, g_5 and g_6, statistically refined as proposed in this work, are shown in Fig. 4 with a Confidence Interval (CI) of 95%, for different values of noise intensity I. Note that the actual values of these coefficients are also depicted in Fig. 4. In this Figure, it is possible to compare the results achieved by the BSA-based and the PSO-based attacks. According with Fig. 4, it is possible to verify that, for all coefficients of $G(z)$, both the BSA-based and PSO-based attacks present good accuracy when $I = 0$ (i.e. without noise, the mean values of the estimated coefficients are close to their actual values). Despite the similar and accurate performance of the two metaheuristics without noise, it is possible to state that the BSA presented a slightly better performance than the PSO in this noise condition ($I = 0$), specially with regard to the coefficients g_1, g_2 and g_3. Note that the performance of the PSO-based attack is degraded when noise is added to the system. This performance degradation of the PSO occurs for $I \geq 0.0025$ and tends to be more expressive with the increase of I. On the other hand, it is possible to verify in Fig. 4 that the BSA-based attack still present good accuracy for noise intensities up to 0.005. When $I \leq 0.005$, all coefficients estimated by the BSA-based attack present a mean close to their actual values and with a small CI. When $I \geq 0.0075$, the performance of the BSA-based attack decreases with the raise of noise in a more expressive way, being at its worst when $I = 0.01$. In general, among the six coefficients of $G(z)$, the estimation of g_2 presents the lowest accuracy for both BSA-based and PSO-based attacks. This behavior is attributed to a lower sensitivity that the output $\hat{y}_a(k)$ of the estimated system has to the variation of g_2. This means that, in this problem, f_j grows faster for errors in g_1, g_3, g_4, g_5 and g_6 than for errors in g_2, making the BSA population converge less accurately in dimension g_2.

The performance of the attacks can also be evaluated in the k domain through the examples provided in Fig. 5, considering two different intensities of noise: without noise, in Fig. 5a; and with $I = 0.005$, in Fig. 5b and c. Figure 5b shows that, without noise, the response of the system estimated by both BSA-based and PSO-based attacks matches the response of the actual system with high accuracy. In Fig. 5b, even with a noise intensity of $I = 0.005$, the response of the system estimated by the BSA-based attack still matches the response of the actual system, indicating the convergence of $G_e(z)$ to $G(z)$ and ratifying the statistics shown in Fig. 4 for the BSA with such noise intensity. On the other hand, when applying the PSO-based attack with the same noise, as exemplified in Fig. 5c, there is a slight difference between the response of the estimated system and the response of the actual system, produced by the mismatch of the estimated coefficients in the presence of such noise intensity. This exemplifies the worse performance of the PSO-based attacks, when compared with the BSA-based attacks, in face of the same noise intensities.



(a) g_1 of $G(z)$ (b) g_2 of $G(z)$ (c) g_3 of $G(z)$

(d) g_4 of $G(z)$ (e) g_5 of $G(z)$ (f) g_6 of $G(z)$

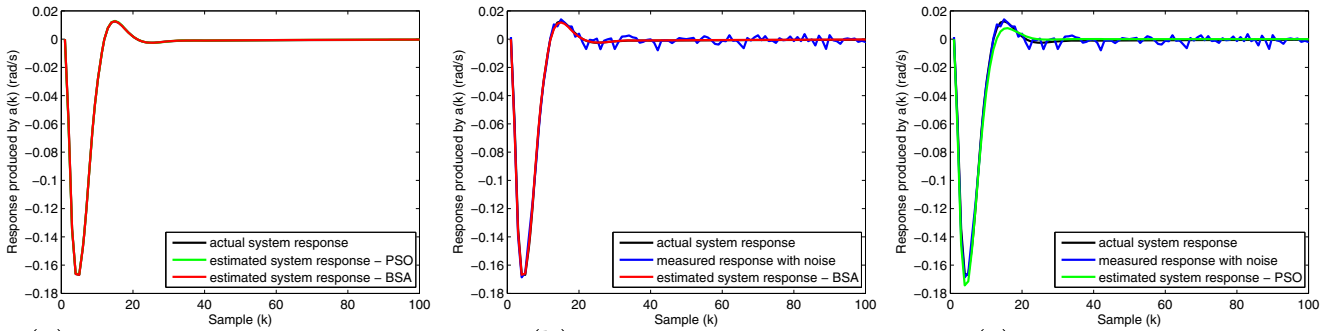
Fig. 4 Mean of the estimated coefficients of $G(z)$, with CI of 95%, in face of different noise intensities I

To synthesize the error of each solution found, $|E_g|$ is computed according to Eq. 15:

$$|E_g| = \sqrt{\sum_{i=1}^{6} (g_i - g_{ei})^2}, \tag{15}$$

wherein g_i and g_{ei} are the actual and estimated coefficients of the attacked system, respectively, and i is the index number of each of the six coefficients of the model being assessed. Note that $|E_g|$ is the module of a vector composed by the error of each coefficient found, which represents another metric to evaluate the performance of each attack.

The histograms of $|E_g|$ are presented in Fig. 6, considering the mentioned noise intensities. It graphically shows that higher values of $|E_g|$ tend to appear more frequently as the noise intensity grows, in both BSA-based and PSO-based attacks. However, based on these histograms it is possible to verify that the mode of $|E_g|$ is close to zero for all noise intensities, using both metaheuristics. This indicates that, even in the presence of noise, most solutions present low deviations from the actual coefficients. Note that, for all noise intensities, the BSA-based attacks provide more results in the modal class – where $|E_g|$ is close to zero – than the PSO-based attacks. Moreover, the worst



(a) BSA and PSO, without noise (b) BSA with $I = 0.005$ (c) PSO with $I = 0.005$

Fig. 5 Response of actual and estimated systems produced by $a(k)$, in face of different noise intensities

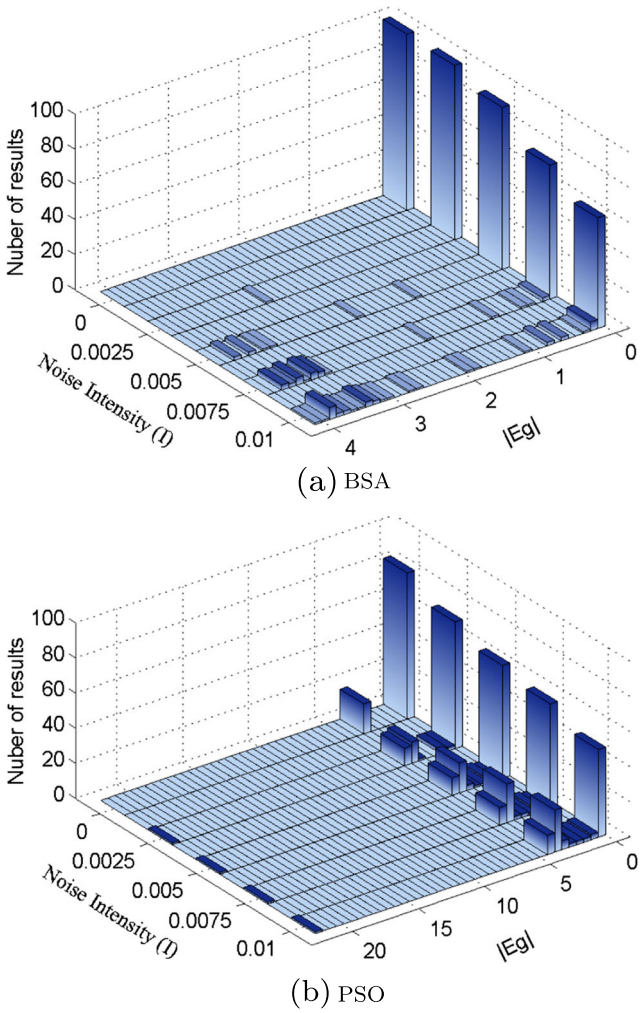


Fig. 6 Histograms of $|E_g|$ for different noise intensities

results of the BSA-based attacks have an $|E_g|$ of about 4 when $I \geq 0.005$, while the worst results of the PSO-based attacks have an $|E_g| > 20$ when $I \geq 0.0025$. These results, together with the statistics shown in Fig. 4, indicate that the performance of the Active System Identification attack is better when implemented with the BSA than with the PSO. It is worth mentioning that, to achieve these results, the BSA-based attacks consumed an average processing time $(6.68 \pm 0.47)\%$ higher than the PSO-based attacks.

In general, the outcomes indicate that, for the same amplitude of attack signal $a(k)$, the performance of the attack tends to decrease as the noise intensity increases (i.e. when the attack signal-to-noise ratio decreases). The minimum length of the attack signal in terms of number of manipulated samples (i.e. one single sample) improves the stealthiness of the attack in the k domain. On the other hand, a minimum attack signal-to-noise ratio required to guarantee the performance of this attack is a drawback with respect to its stealthiness, from the attacker's point of view. This issue makes more difficult for the attacker to approximate the amplitude of $a(k)$ to the noise amplitude or to noise values that have higher probability to occur, which should help to increase the stealthiness of the attack signal in terms of amplitude.

7.2 Data injection attack

The proposed Active System Identification attack is an useful tool – from the attacker point of view – for the design of other sophisticated and accurate attacks. To demonstrate this capability, this section presents a set of data injection attacks, all designed based on the models estimated in Section 7.1 by the Active System Identification attacks. These data injection attacks aim to cause an overshoot of 50% on the rotational speed of the DC motor during its transient response. As mentioned in Section 1, this physically covert interference [7] may cause stress and possibly damages to the plant, reducing its MTBF.

Table 3 Values of a, b and the overshoot obtained with the data injection attacks

		Noise (I) during the system identification attack				
		0	0.0025	0.005	0.0075	0.01
(I)	a	0.25316	0.25485	0.25523	0.58959	0.53297
	b	0.74679	0.74515	0.74477	−0.07354	0.5911
	Overshoot	49.53%	49.49%	49.65%	(*)	(*)
(II)	a	0.25318	0.25286	0.2551	0.27652	0.31407
	b	0.74682	0.74714	0.7449	0.72348	0.68593
	Overshoot	49.52%	49.78%	49.67%	46.91%	42.42%
(III)	a	0.26801	0.32328	0.32816	0.33074	0.33204
	b	0.73199	0.67672	0.67184	0.66926	0.66796
	Overshoot	47.43%	40.70%	40.37%	40.30%	40.38%

(*) The inaccuracy of the data injection attack caused a collateral effect: an expressive steady state error in the motor's rotational speed

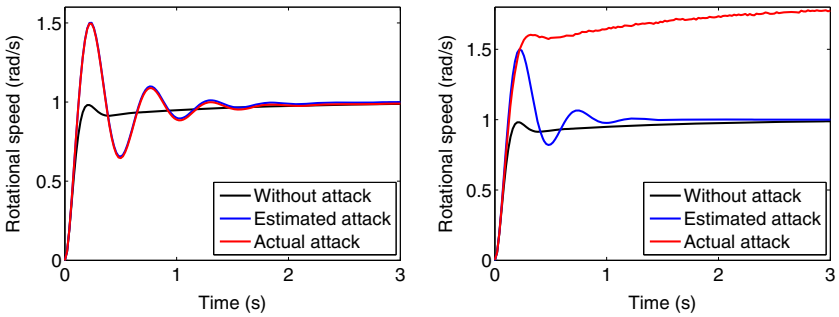


Fig. 7 Data injection attack using models estimated by a BSA-based attack and refined as in [6]

Aware of the estimated model of the NCS, the attacker – acting as an MitM – executes the attack function defined by Eq. 16:

$$y'(k) = ay(k-1) + by'(k-1). \tag{16}$$

wherein a and b are adjusted through a root locus analysis, considering an estimated open-loop transfer function. Note that the attacker is still on the NCS's feedback stream once that, according with Fig. 3, $y(k)$ is the sensor's output and $y'(k)$ is the controller's input.

The models used to design these data injection attacks are built with the mean estimated coefficients shown in Table 2. Note that a and b have to be adjusted for each estimated model which, in turn, vary with the noise condition, the used optimization algorithm and the applied statistical refinement, as shown in Table 2. The values of a and b used in each data injection attack are shown in Table 3, as well as the respective overshoots achieved with the attack. In Table 3, the row (I) contains the data injection attacks designed with the models estimated by the BSA-based attacks using the statistical refinement of [6]. Row (II) contains the data injection attacks designed with the models estimated by the BSA-based attacks using the statistical refinement proposed in this work. As described in Section 7.1, the models estimated in this work and in [6] by the PSO-based attacks do not change due to the statistical refinement method. Thus, in Table 3, the attacks designed with the models estimated by the PSO-based attacks – statistically refined by either of the two methods – are contained in row (III).

Examples of the data injection attacks shown in Table 3 are depicted, in the time domain, in Figs. 7, 8 and 9. In these figures, the curves named as *estimated attack* represent the results predicted by the attacker when applying the designed attack function (16) on the estimated model – i.e. the model provided by the Active System Identification attack. On the other hand, the curves referred as *actual attack* represent the response of the actual system in face of the same attack function (16). In other words, the curve *estimated attack* is the result achieved in a first moment, during the design stage of the attack, and the curve *actual attack* is the result obtained in a second moment, when the designed attack is launched over the actual system.

In rows (I) and (II) of Table 3, it is possible to see that, when $0 \leq I \leq 0.005$, the data provided by the BSA-based Active System Identification attacks produce accurate data injection attacks, either with the statistical refinement of [6] or the statistical refinement proposed in the present work. In these data injection attacks, all overshoots lie between 49.49 and 49.78% – i.e. close to the goal of 50%. However, for $0.0075 \leq I \leq 0.01$, the data injection attacks of row (I) – i.e. using the models estimated by BSA-based attacks and refined as in [6] – produce a collateral behavior on the attacked system. They cause expressive steady state errors in the motor's rotational speed, as indicated, for instance,

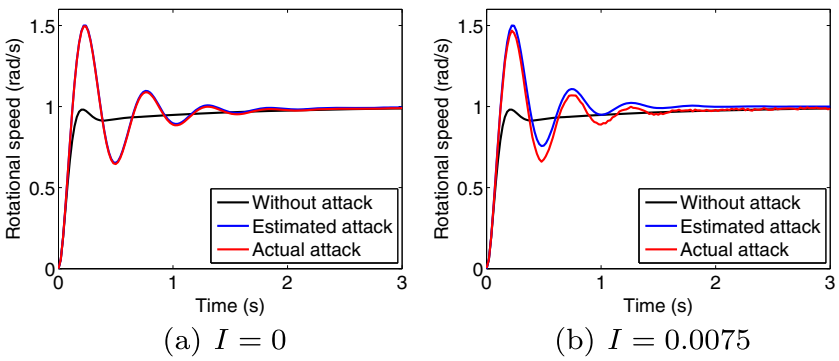


Fig. 8 Data injection attack using models estimated by a BSA-based attack and refined as herein proposed

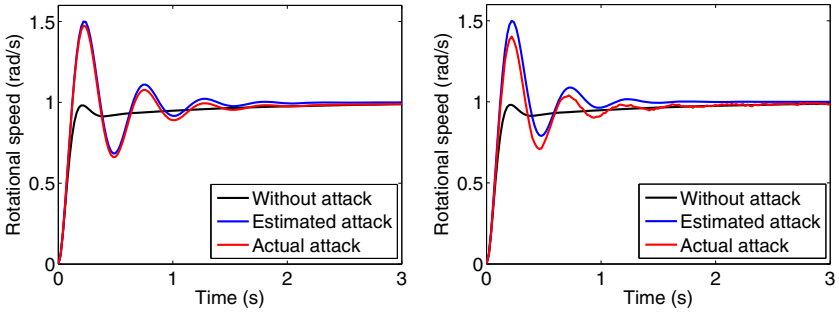


Fig. 9 Data injection attack using models estimated by a PSO-based attack and statistically refined

in Fig. 7b. On the other hand, for $0.0075 \leq I \leq 0.01$, when the statistical refinement proposed in the present work is applied to the BSA-based Active System Identification attacks, the estimated models eliminate the mentioned collateral effects on the data injection attacks. This can be seen in the example shown in Fig. 8b, for $I = 0.0075$, where the response of the actual attack is close to the response of the estimated attack, without a steady state error and with an overshoot of 46.91%. The reason for these different performances is explained by the impact of the statistical refinement in the root locus analysis. When only an outlier coefficient $g_{i,u}$ is eliminated – as in [6] –, instead of eliminating the whole solution S_u from where it belongs – as herein proposed –, the roots of the open-loop transfer function suffer a distortion. For instance, in these simulations, when $0.0075 \leq I \leq 0.01$, the statistical refinement of [6] modifies a pole of $G(z)$ that should be 1. This pole exists due to the use of the PI controller – a premise known by the attacker – and, when modified, influences the adjustment of a and b of Eq. 16. On the other hand, by eliminating the whole solution S_u containing an outlier coefficient $g_{i,u}$, the mean estimated coefficients of $G(z)$ preserve the interdependencies necessary to produce less distorted roots. Note that, as shown in row (III) of Table 3 and in Fig. 9, the PSO-based attacks produce less accurate data injection attacks than the BSA-based attacks statistically refined as proposed in this work. It is worth mentioning that the data injection attacks designed with the models estimated by the PSO-based attacks do not present any collateral effects, using any of the two statistical refinement methods. In both cases, as explained in Section 7.1, the whole solution S_u containing an outlier is eliminated from the set of solutions, producing less distortion in the roots of $G(z)$.

Moreover, with the exception of the attacks of row (I) for $0.0075 \leq I \leq 0.01$, all data injection attacks achieved satisfactory results. However, it is shown that the accuracy of the data injection attack, in general, decreases as the noise intensity increases during the Active System Identification attack.

8 Conclusion

The present work defines and proposes an Active System Identification attack that may be launched over NCSs. The proposed attack is implemented based on two bio-inspired algorithms: the BSA and the PSO. This work demonstrates that the proposed Active System Identification attack is capable to accurately support the design of other sophisticated cyber-physical attacks in NCSs. The results show that the best performance, in general, is achieved by the BSA-based attacks when statistically refined with the method proposed in this paper, specially in the presence of the higher noise intensities.

The capability of the attack to achieve its goal is demonstrated even when: no meaningful information is passing through the NCS's communication links (i.e. when the system had achieved a steady state); the attacker intercepts the communication of the NCS at a single point; and the NCS is noisy.

For future work, we plan to investigate possible techniques to improve the performance of the attack in face of higher noise intensities. Also, we plan – and encourage other researchers – to investigate countermeasures to identify and prevent Active System Identification attacks.

Last, but not least, we plan to improve proposed attack to make it capable to identify systems with uncertain number of unknown coefficients. Preliminary results indicate that, when the number of coefficients is smaller than in the actual system, the algorithm is not able to make the estimated output $\hat{y}_a(k)$ converge to the known $y_a(k)$. In this case, it is not possible to have $\min f_j = 0$, and the global minimum values found by the metaheuristics tends to be high. From the point of view of the attacker, this may be an indicative that the number of coefficients – or dimensions of the metaheuristic – have to be increased, in order to allow $\hat{y}_a(k)$ to match $y_a(k)$. On the other hand, when the number of coefficients is higher than in the actual system, it is possible to have $\min f_j \approx 0$. However, simulations indicate that, even when $\min f_j \approx 0$, the exceeding coefficients does not tend

to 0. In this case, the analysis to eliminate the unnecessary coefficients is not straightforward and still has to be developed, in order to make the algorithm robust to uncertainty with respect to the number of unknown coefficients.

Acknowledgments We appreciate the valuable comments and suggestions of the reviewers that contributed to the great improvement of the original version of this paper.

References

1. Amin S, Litrico X, Sastry S, Bayen AM (2013) Cyber security of water scada systems part i: analysis and experimentation of stealthy deception attacks. IEEE Trans Control Syst Technol 21(5):1963–1970
2. Bou-Harb E, Debbabi M, Assi C (2014) Cyber scanning: a comprehensive survey. IEEE Commun Surv Tutorials 16(3):1496–1519
3. Chen X, Song Y, Yu J (2012) Network-in-the-loop simulation platform for control system. In: Asiasim 2012. Springer, pp 54–62
4. Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219(15):8121–8144
5. Dasgupta S, Routh A, Banerjee S, Agilageswari K, Balasubramanian R, Bhandarkar S, Chattopadhyay S, Kumar M, Gupta A (2013) Networked control of a large pressurized heavy water reactor (phwr) with discrete proportional-integral-derivative (pid) controllers. IEEE Trans Nucl Sci 60(5):3879–3888
6. de Sa AO, da Costa Carmo LFR, Machado RCS (2017) Bio-inspired active attack for identification of networked control systems. In: 10th EAI international conference on bio-inspired information and communications technologies (BICT). ACM, pp 1–8
7. de Sa AO, da Costa Carmo LFR, Machado RCS (2017) Covert attacks in cyber-physical control systems. IEEE Trans Ind Inf 13(4):1641–1651. https://doi.org/10.1109/TII.2017.2676005
8. El-Sharkawi M, Huang C (1989) Variable structure tracking of dc motor for high performance applications. IEEE Trans Energy Convers 4(4):643–650
9. Farooqui AA, Zaidi SSH, Memon AY, Qazi S (2014) Cyber security backdrop: a scada testbed. In: Computing, communications and IT applications conference (comcomap), 2014 IEEE. IEEE, pp 98–103
10. George NV, Panda G (2012) A particle-swarm-optimization-based decentralized nonlinear active noise control system. IEEE Trans Instrum Meas 61(12):3378–3386
11. Guha D, Roy PK, Banerjee S (2016) Application of backtracking search algorithm in load frequency control of multi-area interconnected power system. Ain Shams Eng J
12. Kennedy R, Eberhart JE (1995) Particle swarm optimization. In: Proceedings of 1995 IEEE international conference on neural networks, pp 1942–1948
13. Langner R (2011) Stuxnet: dissecting a cyberwarfare weapon. IEEE Secur Priv 9(3):49–51
14. Long M, Wu C-H, Hung JY (2005) Denial of service attacks on network-based control systems: impact and mitigation. IEEE Trans Ind Inf 1(2):85–96
15. Öncü S, Ploeg J, van de Wouw N, Nijmeijer H (2014) Cooperative adaptive cruise control: network-aware analysis of string stability. IEEE Trans Intell Transp Syst 15(4):1527–1537
16. Precup R-E, Balint A-D, Radac M-B, Petriu EM (2015) Backtracking search optimization algorithm-based approach to pid controller tuning for torque motor systems. In: 2015 9th annual IEEE international systems conference (syscon). IEEE, pp 127–132
17. Sabău Ş, Oară C, Warnick S, Jadbabaie A (2017) Optimal distributed control for platooning via sparse coprime factorizations. IEEE Trans Autom Control 62(1):305–320
18. Shi Y, Huang J, Yu B (2013) Robust tracking control of networked control systems: application to a networked dc motor. IEEE Trans Ind Electron 60(12):5864–5874
19. Si ML, Li HX, Chen XF, Wang GH (2010) Study on sample rate and performance of a networked control system by simulation. In: Advanced materials research, vol 139. Trans Tech Publ, pp 2225–2228
20. Smith R (2011) A decoupled feedback structure for covertly appropriating networked control systems. In: Proceedings of the 18th IFAC world congress 2011, vol 18. IFAC-papersonline
21. Smith RS (2015) Covert misappropriation of networked control systems: presenting a feedback structure. IEEE Control Syst 35(1):82–92
22. Snoeren AC, Partridge C, Sanchez LA, Jones CE, Tchakountio F, Schwartz B, Kent ST, Strayer WT (2002) Single-packet ip traceback. IEEE/ACM Trans Networking (ToN) 10(6):721–734
23. Stallings W (2006) Cryptography and network security: principles and practices. Pearson Education India, Delhi
24. Teixeira A, Shames I, Sandberg H, Johansson KH (2015) A secure control framework for resource-limited adversaries. Automatica 51:135–148
25. Tran T, Ha QP, Nguyen HT (2007) Robust non-overshoot time responses using cascade sliding mode-pid control. Journal of Advanced Computational Intelligence and Intelligent Informatics 11(10):1224–1231
26. Tulleken HJ (1990) Generalized binary noise test-signal concept for improved identification-experiment design. Automatica 26(1):37–49
27. Uong S, Ngamroo I (2015) Coordinated control of dfig wind turbine and svc for robust power system stabilization. In: 2015 12th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON). IEEE, pp 1–6